NARATIF(Jurnal Ilmiah Nasional Riset Aplikasi dan Teknik Informatika)
Vol. 04 No. 01 Juni 2022
P-ISSN: 2656-7377 || E-ISSN: 2714-8467

PENGUJIAN *WHITE-BOX* PADA APLIKASI *DEBT MANAGER* BERBASIS ANDROID

Harya Gusdevi[1], Sri Kuswayati[2], Muhammad Iqbal[3], Mohamad Fikri Abu Bakar[4],
Nesha Novianti[5], Rizky Ramadan[6]
Program Studi Teknik Informatika[1,2,3,4,5,6]
Sekolah Tinggi Teknologi Bandung[1,2,3,4,5,6]
deviharya@gmail.com[1], srikuswayati@sttbandung.ac.id[2], iqbal.xtab404@gmail.com[3], fikriabu40@gmail.com[4],
nesha@gmail.com[5], rizki123304@gmail.com[6]

Abstrak

Semakin berkembangnya ilmu pengetahuan dan teknologi pada zaman sekarang, telah banyak menciptakan berbagai perangkat seperti komputer dan *mobile*. Setiap perangkat komputer dan *mobile* terdapat aplikasi didalamnya, aplikasi yang dibangun inilah tidak lepas dari proses pengujian. Sebelum aplikasi ini dapat digunakan oleh masyarakat tentu perlu diuji terlebih dahulu mengenai kelayakannya untuk digunakan oleh masyarakat luas. Proses atau metode dalam pengujian pada tiap aplikasi pun berbeda-beda. Ada yang menggunakan Teknik *black-box testing, white-box testing* dan lainnya. Penelitian ini, pengujian pada aplikasi *Debt Manager* akan digunakan dengan Teknik *white-box testing* yang bertujuan untuk menguji perangkat lunak dengan cara menganalisa dan meneliti struktur internal dan kode dari perangkat lunak. Maka dari itu dalam jurnal ini akan dijelaskan bagaimana proses pengujian *white-box* pada aplikasi *Debt Manager*. Pengujian dengan menggunakan *white-box testing* ini dilakukan dengan cara menguji di bagian fungsionalitas seperti menu buat utang, bayar utang dan tambah pinjaman. Pengujian dilakukan pada masing-masing bagian fungsionalitas dengan cara pengujian terhadap *path*, pengujian untuk menilai tingkat resiko dari prosedur aplikasi dan pengujian *test case*. Hasil dari pengujan ketiga bagian fungsionalitas menu buat utang, bayar utang dan tambah pinjaman adalah hasil valid pada tiap masing-masing tabel menyatakan tidak ditemukan adanya *error* dengan kompleksitas yang rendah karena hanya 2-3 jalur yang ada pada setiap fungsi.
Kata kunci : Utang, Piutang, Aplikasi, *White-Box*

Abstract

The development of science and technology in this day and age, has created many devices such as computers and mobile. Every computer and mobile device has an application in it, this built application cannot be separated from the testing process. Before this application can be used by the public, of course, it needs to be tested first regarding its feasibility for use by the wider community. The process or method of testing each application is also different. Some use black-box testing techniques, white-box testing and others. This research, testing on the Debt Manager application will be used with white-box testing techniques which aim to test software by analyzing and researching the internal structure and code of the software. Therefore, in this journal, it will be explained how the white-box testing process on the Debt Manager application. Testing using white-box testing is carried out by testing in the functionality section such as menus to create debts, pay debts and add loans. Testing is carried out in each part of the functionality by testing the path, testing to assess the level of risk of the application procedure and testing the test case. The result of the third part of the menu functionality create debt, pay debt and add loans is a valid result in each table stating that there are no errors found with low complexity because only 2-3 paths exist in each function.
Keywords :Debt, Receivable, Application, White-Box

I. PENDAHULUAN

Semakin berkembangnya ilmu pengetahuan dan teknologi pada masa zaman sekarang ini, telah banyak menciptakan berbagai perangkat dengan fungsi dan manfaat yang memudahkan bagi setiap orang yang menggunakanannya. Terlebih lagi setiap orang memiliki dengan banyaknya perangkat seperti komputer, salah satu perangkat komputer lainya yaitu perangkat *mobile phone*, kualitas dan efektifitas dalam bekerja semakin meningkat semenjak adanya perangkat komputer dan *mobile*. *Mobile Phone*(*smartphone*) menawarkan kemudahan melalui aplikasi untuk membantu manusia dalam menyelesaikan pekerjaan dengan cepat, dan *Mobile Phone* ini dapat membantu aktifitas lain seperti untuk berkomunikasi. Teknologi *Mobile Phone* ini sudah dapat terhubung jaringan *Wi-Fi* yang semakin memudahkan penggunaannya[1].

Setiap banyak orang-orang yang terlibat utang-piutang, baik itu utang ke keluarga, sodara, teman, sahabat, bahkan utang ke bank. Seringkali kita sebagai manusia lupa akan kewajiban kita untuk membayar utang, tidak sedikit pula orang yang hanya sebatas meminjam namun tidak dikembalikan lagi. Utang ialah memberikan harta kepada orang yang akan memanfaatkannya dan mengembalikan gantinya dikemudian hari dengan jumlah yang sama[2].

Pengujian ke *white-box testing* adalah metode *white-box* bisa disebut dengan pengujian yang terstruktur, pengujian *transparent box*, pengujian berdasarkan logika atau pengujian berdasarkan kode. Kata *white-box* yang berarti kotak putih/transparan memiliki arti pada sebuah metode *test case*, Sistem yang akan di uji diumpamakan sebagai suatu kotak (*box*), dan kata *white/transparent* mengacu pada kotak itu yang terlihat jelas isinya Metode pengujian pada *white-box testing* ini seringkali dilakukan untuk memberikan dan membuat suatu jaminan bahwa seluruh jalur-jalur yang independen hanya menggunakan modul yang biasanya minimal satu kali, Keputusan yang sifatnya logis dapat digunakan di semua kondisi *true* (benar) atau *false* (salah). Mengeksekusi seluruh perulangan yang ada ke pada batas nilai dan operasional di setiap situasi dan kondisi, Syarat yang dilakukan dalam menjalankan strategi *white-box testing*, Mendefinisikan tentang seluruh alur-alur logika yang ada, Membangun dan membuat suatu kasus yang akan digunakan untuk tahap pengujian.

Aplikasi *Debt Manager* akan dilakukan pengujian perangkat lunak dimana tujuannya untuk mengetahui jika ada kesalahan, yang menyebabkan kegagalan *system* atau *error*. Aplikasi *Debt Manager* ini melakukan Pengujian dengan menggunakan metode *white-box*. Penguji dapat melihat sistem dan cara kerja dari suatu perangkat lunak. Tujuan Aplikasi *Debt Manager* dilakukan pengujian *White-box* adalah untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang dibuat ada yang salah atau tidak jadi dengan memeriksa komponen perangkat lunak apakah berjalan semestinya dengan melihat internal kode (*source code*) dari perangkat lunak tersebut. Dengan menggunakan metode pengujian *white-box* maka alur perangkat lunak dapat diuji.

Berdasarkan latar belakang tersebut maka aplikasi *Debt Manager* akan dilakukan pengujian *white-box testing* yang dimana pada aplikasi ini akan diuji menggunakan metode *white-box testing* agar hasil pengujian yang telah didapatkan akan dilakukan evaluasi kembali, sehingga mampu menyimpulkan apakah perangkat lunak sudah berjalan sesuai dengan kebutuhan yang diharapkan atau sebaliknya.

II. TINJAUAN PUSTAKA

1. Android

Android merupakan sebuah *system* operasi telepon berbasis Linux untuk telepon seluler seperti *smartphone* dan komputer atau tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi ini digunakan oleh berbagai peranti bergerak [3]. Ahli lain mengatakan bahwa android merupakan suatu sistem operasi yang menggunakan java sebagai bahasa pemrograman dan berbasis linux serta dirancang khusus untuk telepon seluler layar sentuh seperti *smartphone* dan tablet [4].

2. Aplikasi *Mobile*

Aplikasi *mobile* yaitu sebuah perangkat lunak yang dirancang untuk dapat berjalan pada perangkat bergerak contohnya yaitu seperti *smartphone*. *Smartphone* berperan sebagai *high end mobile phone* yang dilengkapi dengan kemampuan *mobile computing*. Dengan kemampuannya tersebut, pengguna dapat dengan mudah menambahkan aplikasi dan menambahkan fungsi-fungsi sesuai kebutuhan pengguna dengan performa yang cukup tinggi. Kemudahan dalam pengoperasian aplikasi *mobile* juga menjadi salah satu faktor yang dapat menyebabkan peningkatan yang semakin meningkatnya pengguna *smartphone* dalam satu tahun terakhir [5].

3. *White-box Testing*

White-box testing atau pengujian *white-box* dilakukan untuk menguji dan menganalisis kode program bilamana terjadi kesalahan atau tidak di sebut dengan pengujian *white box* [6]. Terdapat pendapat lain mengenai pengertian dari pengujian *white box* ini dilakukan dengan melihat *pure kode* tanpa melihat tampilan *interface* dari halaman aplikasi [7]. *White Box* sendiri mempunyai beberapa teknik di dalam pengujiannya, seperti : *Data Flow Testing, Control Flow Testing, Basic Path / Path Testing*, dan *Loop Testing* [8].

Kelebihan dari penggunaan metode *white-box testing* adalah dapat memperlihatkan galat pada kode yang dibuat dengan menghapus baris yang tidak diperlukan serta maksimalnya cakupan pengujian aplikasi saat uji coba sebuah *scenario* [9].

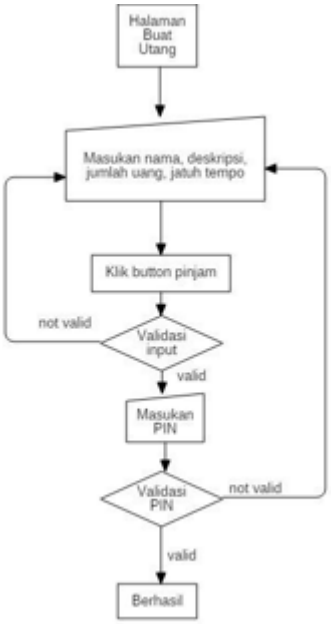
III. PEMBAHASAN

Berikut merupakan pembahasan dari beberapa bagian fungsional dari Aplikasi *Debt Manager* yang akan diuji dengan menggunakan *white-box testing*.

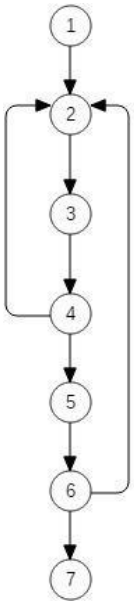
1. Buat Utang

```
ContainerButtonProgress(
  widget: ProgressButtonCustom(
    colorIdle: Variables.primaryColor,
    colorSuccess: Variables.colorGreen,
    idleText: 'Pinjam',
    idleIcon: Icon(Icons.send_outlined, color: Colors.white),
    failText: 'Pinjam Failed',
    failIcon: Icon(Icons.cancel_outlined, color: Colors.white),
    successText: 'Pinjam Success',
    successIcon:
        Icon(Icons.check_circle_outline, color: Colors.white),
    onPressed: () async {
      FocusScope.of(context).unfocus();
      setState(() => loading = ButtonState.loading);
      String strTanggalPinjam = now.toString();
      DateTime pinjam = DateTime.parse(strTanggalPinjam);
      setState(() {
        tanggalPinjam = pinjam.microsecondsSinceEpoch;
      });
      if (jatuhTempo == null) {
        String strJatuhTempo = now.toString();
        DateTime tempo = DateTime.parse(strJatuhTempo);
        setState(() {
          jatuhTempo = tempo.microsecondsSinceEpoch;
        });
      }
      if (_formKey.currentState.validate()) {
        pin = await showPinDialog(
            context,
            Variables.primaryColor,
            Variables.colorGreen,
            (val) => {},
            pin,
            'Enter Your PIN');
        pinFromDB =
            await DatabaseService(uid: user.uid).getPin();
        if (pin == pinFromDB) {
          dynamic result = await DatabaseService()
              .createUtangOrPiutang(
                  namaLengkap,
                  deskripsi,
                  int.parse(jumlahUang),
                  tanggalPinjam,
                  jatuhTempo,
                  'utang');
          if (result == null) {
            setState(() => loading = ButtonState.fail);
          }
          setState(() => loading = ButtonState.success);
          await Future.delayed(Duration(seconds: 1));
          setState(() => loading = ButtonState.idle);
          Navigator.pushReplacement(
              context,
              MaterialPageRoute(
                  builder: (context) => BottomNavigation()));
        } else {
          setState(() => loading = ButtonState.fail);
          await Future.delayed(Duration(seconds: 1));
          setState(() => loading = ButtonState.idle);
        }
      } else {
        setState(() => loading = ButtonState.fail);
        await Future.delayed(Duration(seconds: 1));
        setState(() => loading = ButtonState.idle);
      }
    },
    stateButton: loading,
  ),
)
```

Gambar 1. *Coding* Buat Utang



Gambar 2 *Flowchart* Buat Utang



Gambar 3 *Flowgraph* Buat Utang

V(G)= (E-N) +2
V(G)= Jumlah Region
E= Jumlah edge yang ditentukan dengan gambar panah N= Jumlah simpul grafik (node) dengan gambar lingkaran

V(G)= (8-7) +2
V(G)= 3

Jalur 1= 1-2-3-4-5-6-7
Jalur 2= 1-2-3-4-2-3-4-5-6-7
Jalur 3= 1-2-3-4-5-6-2-3-4-5-6-7

TABEL I
PENGUJIAN *PATH* FUNGSI BUAT UTANG

Path	1
Jalur	1-2-3-4-5-6-7
Skenario	1 Halaman buat utang 2 Masukan nama, deskripsi, jumlah uang, jatuh tempo3 Klik *button* pinjam 4 Validasi input (valid)5 Masukan PIN 6 Validasi PIN (valid)7 Berhasil
Hasil Pengujian	Berhasil
Path	2
Jalur	1-2-3-4-2-3-4-5-6-7
Skenario	1 Halaman buat utang 2 Masukan nama, deskripsi, jumlah uang, jatuh tempo3 Klik *button* pinjam 4 Validasi input (*not valid*) 2 Masukan nama, deskripsi, jumlah uang, jatuh tempo3 Klik *button* pinjam 4 Validasi input (valid)5 Masukan PIN 6 Validasi PIN (valid) 7 Berhasil
Hasil Pengujian	Berhasil

Path	3
Jalur	1-2-3-4-5-6-2-3-4-5-6-7
Skenario	1 Halaman buat utang 2 Masukan nama, deskripsi, jumlah uang, jatuh tempo3 Klik *button* pinjam 4 Validasi input (valid)5 Masukan PIN 6 Validasi PIN (*not valid*) 2 Masukan nama, deskripsi, jumlah uang, jatuh tempo3 Klik *button* pinjam 4 Validasi input (valid)5 Masukan PIN 6 Validasi PIN (valid)7 Berhasil
Hasil Pengujian	Berhasil

Setelah diketahui jumlah jalur independennya, maka akan dilakukan perbandingan menggunakan tabel hubungan antara *cyclomatic complexity* dan resiko pada tabel berikut:

TABEL II
RESIKO DARI TINGKAT PROSEDUR APLIKASI

Nilai	Tipe Prosedur	Tingkat Resiko
1-4	Prosedur Sederhana	Rendah
5-10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11-20	Prosedur yang lebih kompleks	Menengah
21-50	Prosedur yang kompleks dan kritis	Menengah
>50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi

Jadi menurut tabel di atas untuk fungsi buat utang ini memiliki resiko yang rendah dengan tingkat prosedur yang sederhana karena memiliki jalur independen berjumlah 3. Kemudian, setelah diketahui jalur independennya maka langkah selanjutnya adalah *test case*, seperti tabel di bawah ini:

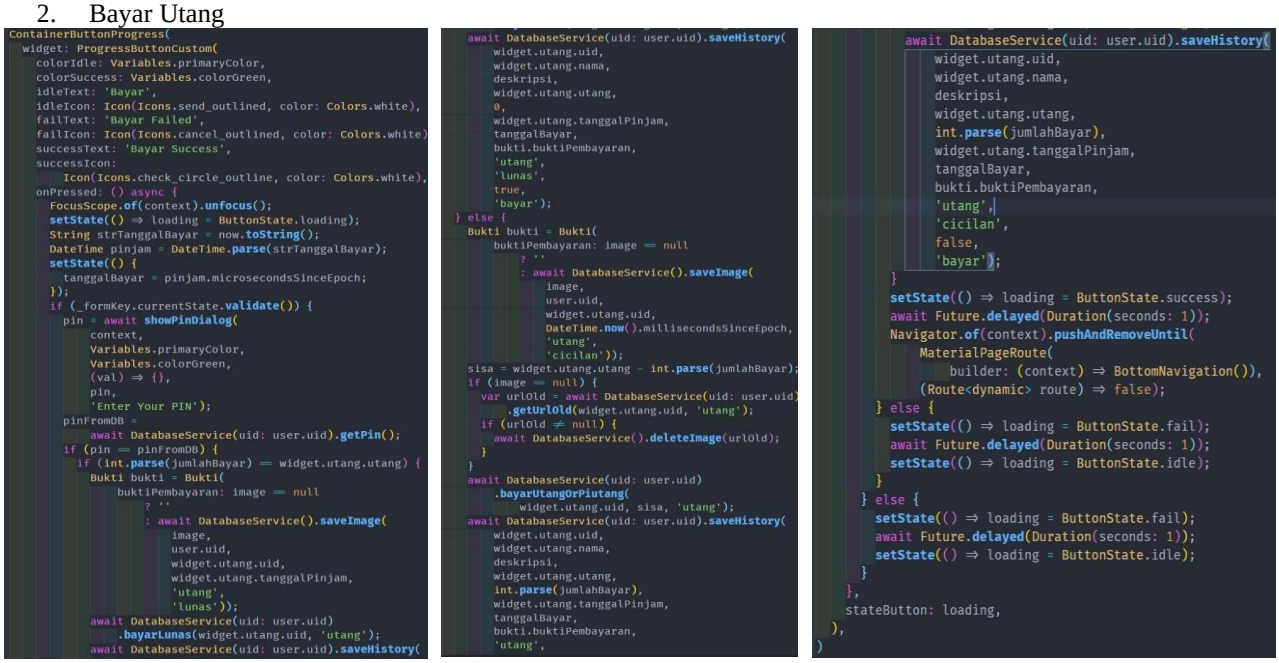
TABEL III
TEST CASE BUAT UTANG

No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	Uji Buat Utang 1	*User* memilih halaman buat utang dan memasukan nama, deskripsi,jumlah uang, jatuh tempo, kemudian *user* memasukan PIN yang valid	Berhasil menambahkan utang dan data berhasil ditambahkan ke *database*	Sistem menampilkan pesan sukses menambahkan utang dan kembali ke halaman utama	Valid
2	Uji Buat Utang 2	*User* memilih halaman buat utang dan tidakmemasukan semua kolom input kemudian *user* mengklik tombol pinjam, lalu *user* memauskan kembali data input, kemudian *user* memasukanPIN yang valid	Sebelum data berhasil ditambahkan ke *database*, *user* mendapatkan pesan *error* karena tidak mengisi semua kolom input. Kemudian setelah berhasilinput dan memasukan PINmaka data berhasil ditambahkan ke *database*	Sistem menampilkan pesan *error* terlebih dahulukarena semua kolom input tidak diisi, baru menampilkan pesan sukses setelah berhasilsemua validasi	Valid

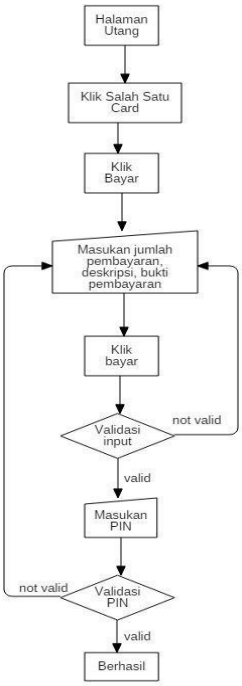
No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
3	Uji Buat Utang 3	*User* memilih halaman buat utang dan memasukan semua data input kemudian *user* mengklik *button* pinjam namun salah memasukan PIN, lalu *user* mengulangi mengklik tombol pinjam dan memasukan PIN yang benar	Muncul pesan *error* ketika salah memasukan PIN, dan berhasil ketika *user* sudah memasukan PIN yang benar	Sistem menampilkan pesan *error* ketika *user* salah memasukan PIN dan berhasil memasukan data ke *database* ketika *user* sudah berhasil memasukan PIN yang benar	Valid

Pada tabel III yang dibuat berdasarkan jalur independen yang telah dibuat dan telah didapatkan hasil valid untuk ketiga pengujian yang dilakukan sehingga tidak ditemukan *error*.

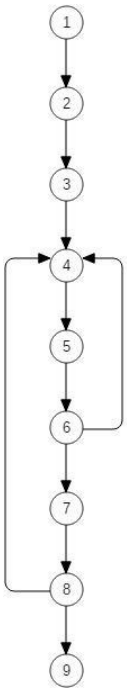
2. Bayar Utang



Gambar 4. *Coding* Bayar Utang



Gambar 5. *Flowchart* Bayar Utang



Gambar 6. *Flowgraph* Bayar Utang

V(G)= (E-N) +2
V(G)= Jumlah Region
E= Jumlah edge yang ditentukan dengan gambar panah N= Jumlah simpul grafik (node) dengan gambar lingkaran

V(G)= (10-9) +2
V(G)= 3

Jalur 1= 1-2-3-4-5-6-7-8-9
Jalur 2= 1-2-3-4-5-6-4-5-6-7-8-9
Jalur 3= 1-2-3-4-5-6-4-5-6-7-8-4-5-6-7-8-9

TABEL IV
PENGUJIAN *PATH* FUNGSI BAYAR UTANG

Path	1
Jalur	1-2-3-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik bayar 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (valid)9 Berhasil
Hasil Pengujian	Berhasil
Path	2
Jalur	1-2-3-4-5-6-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik bayar 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (*not valid*) 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (valid)9 Berhasil
Hasil Pengujian	Berhasil

Path	3
Jalur	1-2-3-4-5-6-4-5-6-7-8-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik bayar 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (*not valid*) 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (not valid) 4 Memasukan jumlah pembayaran, deskripsi, bukti pembayaran5 Klik bayar 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (valid)9 Berhasil
Hasil Pengujian	Berhasil

Setelah diketahui jumlah jalur independennya, maka akan dilakukan perbandingan menggunakan tabel hubungan antara *cyclomatic complexity* dan resiko pada tabel berikut:

TABEL V
RESIKO DARI TINGKAT PROSEDUR APLIKASI

Nilai	Tipe Prosedur	Tingkat Resiko
1-4	Prosedur Sederhana	Rendah
5-10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11-20	Prosedur yang lebih kompleks	Menengah
21-50	Prosedur yang kompleks dan kritis	Menengah
>50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi

Jadi menurut tabel di atas untuk fungsi buat utang ini memiliki resiko yang rendah dengan tingkat prosedur yang sederhana karena memiliki jalur independen berjumlah 3. Kemudian, setelah diketahui jalur independennya maka langkah selanjutnya adalah *test case*, seperti tabel di bawah ini:

TABEL VI
TEST CASE BAYAR UTANG

No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	Uji Bayar Utang1	*User* melakukan bayar utang dengan memasukan jumlah pembayaran, deskripsi, dan bukti pembayaran, kemudian memasukan PIN yang valid	Berhasil membayar utang dan data disimpan ke *database* dengan memunculkan pesan sukses	Sistem menampilkan pesan sukses bayar utang	Valid
2	Uji Bayar Utang2	*User* melakukan bayar utang dengan tidak memasukan datainput untuk pembayaran saat pertama kali, kemudian memasukan jumlah pembayaran, deskripsi, dan bukti pembayaran, kemudian memasukan PIN yang valid	Berhasil membayar utang dengan pesan pertama adalah pesan *error* input kosong dan pesan kedua sukses dan data berhasil ditambahkan ke *database*	Sistem menampilkan pesan *error* saat *user* tidak mengisi kolom input, dan berhasil menambahkan data ke *database* ketika *user* sudah mengisi semua input dan PIN yang valid	Valid

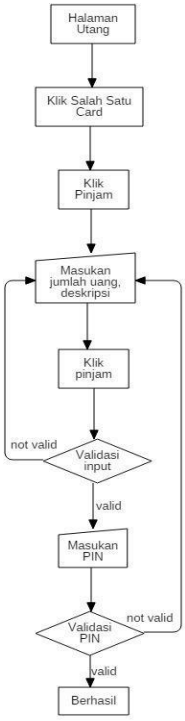
No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
3	Uji Bayar Utang 3	*User* melakukan bayar utang dengan memasukan jumlah pembayaran, deskripsi, dan bukti pembayaran, kemudian memasukan PIN yang salah lalu *user* mengulangi memasukan PINyang valid	Berhasil membayar utang dengan pesan pertama adalah pesan *error* PIN not valid dan pesan kedua sukses dan data berhasil ditambahkan ke *database*	Sistem menampilkan pesan *error* saat *user* salah memasukan PIN yang pertama, dan memunculkan pesan sukses ketika *user* berhasil memasukan PIN yang valid dan data berhasil dimasukan ke *database*	Valid

Pada tabel *test case* diatas yang dibuat berdasarkan jalur independen yang telah dibuat dan telah didapatkan hasil valid untuk ketiga pengujian yang dilakukan sehingga tidak ditemukan *error*.

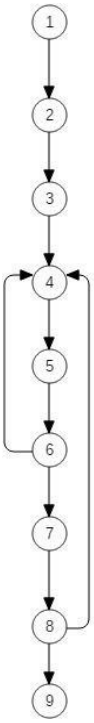
3. Tambah Pinjaman

```
ContainerButtonProgress(
  widget: ProgressButtonCustom(
    colorIdle: Variables.primaryColor,
    colorSuccess: Variables.colorGreen,
    idleText: 'Pinjam',
    idleIcon: Icon(Icons.send_outlined, color: Colors.white),
    failText: 'Pinjam Failed',
    failIcon: Icon(Icons.cancel_outlined, color: Colors.white),
    successText: 'Pinjam Success',
    successIcon:
        Icon(Icons.check_circle_outline, color: Colors.white),
    onPressed: () async {
      FocusScope.of(context).unfocus();
      setState(() => loading = ButtonState.loading);
      if (_formKey.currentState.validate()) {
        pin = await showPinDialog(
            context,
            Variables.primaryColor,
            Variables.colorGreen,
            (val) => {},
            pin,
            'Enter Your PIN');
        pinFromDB =
            await DatabaseService(uid: user.uid).getPin();
        if (pin == pinFromDB) {
          tambahUtang = widget.utang.utang + int.parse(utang);
          await DatabaseService(uid: user.uid)
              .tambahUtangOrPiutang(
                  widget.utang.uid,
                  tambahUtang,
                  deskripsi,
                  'utang',
                  DateTime.now().microsecondsSinceEpoch,
                  int.parse(utang),
                  'tambah');
          setState(() => loading = ButtonState.success);
          await Future.delayed(Duration(seconds: 1));
          Navigator.of(context).pushAndRemoveUntil(
              MaterialPageRoute(
                  builder: (context) => BottomNavigation()),
              (Route<dynamic> route) => false);
        } else {
          setState(() => loading = ButtonState.fail);
          await Future.delayed(Duration(seconds: 1));
          setState(() => loading = ButtonState.idle);
        }
      } else {
        setState(() => loading = ButtonState.fail);
        await Future.delayed(Duration(seconds: 1));
        setState(() => loading = ButtonState.idle);
      }
    },
    stateButton: loading,
  ),
)
```

Gambar 7. *Coding* Tambah Pinjaman



Gambar 8. *Flowchart* Tambah Pinjaman



Gambar 9. *Flowgraph* Tambah Pinjaman

V(G)= (E-N) +2
V(G)= Jumlah Region
E= Jumlah edge yang ditentukan dengan gambar panah N= Jumlah simpul grafik (node) dengan gambar lingkaran

V(G)= (10-9) +2
V(G)= 3

Jalur 1= 1-2-3-4-5-6-7-8-9
Jalur 2= 1-2-3-4-5-6-4-5-6-7-8-9
Jalur 3= 1-2-3-4-5-6-7-8-4-5-6-7-8-9

TABEL VII
PENGUJIAN PATH FUNGSI TAMBAH PINJAMAN

Path	1
Jalur	1-2-3-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik pinjam 4 Masukan jumlah uang, deskripsi5 Klik pinjam 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (valid)9 Berhasil
Hasil Pengujian	Berhasil
Path	2
Jalur	1-2-3-4-5-6-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik pinjam 4 Masukan jumlah uang, deskripsi5 Klik pinjam 6 Validasi input (*not valid*) 4 Masukan jumlah uang, deskripsi5 Klik pinjam 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (valid)9 Berhasil
Hasil Pengujian	Berhasil

Path	3
Jalur	1-2-3-4-5-6-7-8-4-5-6-7-8-9
Skenario	1 Halaman utang 2 Klik salah satu card3 Klik pinjam 4 Masukan jumlah uang, deskripsi5 Klik pinjam 6 Validasi input (valid)7 Masukan PIN 8 Validasi PIN (not valid) 4 Masukan jumlah uang, deskripsi5 Klik pinjam 6 Validasi input (valid) 7 Masukan PIN 8 Validasi PIN (valid) 9 Berhasil
Hasil Pengujian	Berhasil

Setelah diketahui jumlah jalur independennya, maka akan dilakukan perbandingan menggunakan tabel hubungan antara *cyclomatic complexity* dan resiko pada tabel berikut:

TABEL IX
RESIKO DARI TINGKAT PROSEDUR APLIKASI

Nilai	Tipe Prosedur	Tingkat Resiko
1-4	Prosedur Sederhana	Rendah
5-10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11-20	Prosedur yang lebih kompleks	Menengah
21-50	Prosedur yang kompleks dan kritis	Menengah
>50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi

Jadi menurut tabel di atas untuk fungsi buat utang ini memiliki resiko yang rendah dengan tingkat prosedur yang sederhana karena memiliki jalur independen berjumlah 3. Kemudian, setelah diketahui jalur independennya maka langkah selanjutnya adalah *test case*, seperti tabel di bawah ini:

TABEL X
TEST CASE TAMBAH PINJAMAN

No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
1	Uji Tambah Pinjaman 1	*User* melakukan tambah pinjaman dengan memasukan jumlah uang, deskripsi dan mengklik *button* pinjam dan memasukan PINyang valid	Berhasil tambah pinjaman dan data berhasil ditambahkan ke *database* dengan pesan sukses berhasil tambah pinjaman	Sistem menampilkanpesan sukses dan data berhasil disimpan ke *database*	Valid
2	Uji Tambah Pinjaman 2	*User* melakukan tambah pinjaman dengan tidak memasukan input kemudian mengklik *button* pinjam. Lalu *user* memasukan ulang data di kolom input dan mengklik *button* pinjam dan memasukan PIN yang valid	Muncul pesan *error* ketika *user* mengklik *button* pinjam tanpa ada kolom input yang diisi, kemudian setelah *user* mengisi semua kolom input maka berhasil tambah pinjaman dan data berhasil ditambahkan ke *database* dengan pesan sukses berhasil tambah pinjaman	Sistem menampilkan pesan *error* terlebih dahulu karena *user* tidak mengisi semua kolom input. Lalu ketika *user* sudah mengisi semua kolom sistem menampilkan pesan sukses dan data berhasil disimpan ke *database*	Valid

No	Nama Skenario	Kegiatan	Hasil yang diharapkan	Hasil	Keterangan
3	Uji Tambah Pinjaman 3	*User* melakukan tambah pinjaman dengan memasukan jumlah uang, deskripsi dan mengklik *button* pinjam dan memasukan PIN yang tidak valid kemudian *user* mengklik ulang *button* pinjam dan memasukan PIN yang valid	Muncul pesan *error* ketika *user* mengklik *button*pinjam dan salah memasukan PIN, kemudian setelah *user* berhasil memasukan PIN yang valid maka berhasil tambah pinjaman dan data berhasil ditambahkan ke *database* denganpesan sukses berhasil tambah pinjaman	Sistem menampilkan pesan *error* terlebih dahulu karena *user* salah memasukan PIN. Lalu ketika *user* berhasil memasukan PIN yang valid maka sistem menampilkan pesan sukses dan data berhasil disimpan ke *database*	Valid

Pada tabel *test case* diatas yang dibuat berdasarkan jalur independen yang telah dibuat dan telah didapatkan hasil valid untuk ketiga pengujian yang dilakukan sehingga tidak ditemukan *error*.

IV. KESIMPULAN

Berdasarkan hasil penelitian mengenai pengujian *white-box* untuk aplikasi *Debt Manager* telah dinyatakan lolos tanpa adanya *error* untuk fungsionalitas utamanya, terlihat pada pengujian bagian buat utang, bayar utang, dan tambah pinjaman semuanya dinyatakan valid dengan kompleksitas yang rendah karena hanya 2-3 jalur yang ada pada setiap fungsi.

REFERENSI

[1] Jery Jordan RSH. Aplikasi Manajemen Hutang Piutang Untuk Usaha Kecil Menengah Berbasis Android. J Mhs Fak Tek dan Ilmu Komput. 2020;1(1):439–50.
[2] A. b. Muhammad, Ensiklopedi Fiqih Muamalah dalam Pandangan 4 Madzhab, Yogyakarta: Maktabah Al-Hanif, 2017.
[3] Syarwan BA, Purba KR, Setiawan A. Pembuatan Aplikasi Management Keuangan Pribadi Berbasis Android. J Infra Petra. 2018;3–6.
[4] Jubilee , Enterprise (2015), Mengenal Pemograman *Database*, Jakarta: PT Elex Media Komputindo.
[5] Seputra KA, Sandiasa G. Rancang Bangun Sistem Informasi Satgas Gotong Royong (Si Garong) Desa Adat Berbasis *Mobile*. J Nas Pendidik Tek Inform. 2020;9(3):338.
[6] E. S. Eriana, G. Saputri, and J. Rachmansyah, "PENGUJIAN WHITE BOX APLIKASI PEMESANAN AKSESORIS MOTOR BERBASIS WEB."
[7] R. Mahmood, N. Esfahani, T. Kacem, N. Mirzaei, S. Malek, and A. Stavrou, "A whitebox approach for automated security testing of Android applications on the cloud," 2012,
[8] V. P. Katiyar and S. Patel, "White-Box Testing Technique for Finding Defects," Glob. J. Res. Anal., vol. 8, no. 7, pp. 83–85, 2019, [Online]
[9] K. Mohd. Ehmer and K. Farmeena, "A Comparative Study of White Box , Black Box and Grey Box Testing Techniques," Int. J. Adv. Comput. Sci. Appl., vol. 3, no. 6, pp. 12– 15, 2012.