Jurnal Informatika Universitas Pamulang **ISSN: 2541-1004**
Penerbit: Program Studi Teknik Informatika Universitas Pamulang **e-ISSN: 2622-4615**
Vol. 5, No. 2, Juni 2020 (111-119) **10.32493/informatika.v5i2.4713**

Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path

Citra Teguh Pratala[1], Ebenhaezer Mahardhika Asyer[2], Ima Prayudi[3], Aries Saifudin[4]

Teknik Informatika, Universitas Pamulang, Jl. Raya Puspitek No.46, Buaran, Serpong, Tangerang Selatan, Banten, Indonesia, 15310
E-mail: [1]c.teguhpratala@gmail.com, [2]ebenhaezer289@gmail.com, [3]prayudi.ima@gmail.com, [4]aries.saifudin@unpam.ac.id

Submitted Date: March 28th, 2020 Reviewed Date: April 05th, 2020
Revised Date: April 06th, 2020 Accepted Date: June 29th, 2020

Abstract

An application that has poor quality and defects not only causing additional time of software development process but also can deliver disadvantage to the users of the application. Every application will not be separated from the testing process. The testing process aims to find out errors in the software. One of the methods used is white box testing. White box testing is a testing method at the level of the software flow. To find out whether the Cash Flow application does not have an error it will be tested with a base path technique that starts from making a Flow Graph, Cyclomatic Complexity (CC) and doing a Case unit. The implementation of the white box testing method using the Base path technique in the cash flow application can evaluate the complexity of the program flow and while conducting unit tests it can determine the number of test scenarios to be performed. After doing Cash Flow application testing with no errors, thus all tests were successful. The results of this test can be used as a reference to improve the application as expected.

Keywords: White Box; Software Testing; Android; Junit; Basis Path

Abstrak

Aplikasi yang memiliki kualitas buruk dan cacat bukan hanya mengakibatkan penambahan waktu pengembangan perangkat lunak tetapi dapat menyebabkan kerugian bagi pengguna aplikasi tersebut. Setiap aplikasi tidak akan lepas dari proses pengujian. Proses pengujian bertujuan untuk mengetahui galat pada perangkat lunak. Salah satu metode yang digunakan adalah *white box testing*. *White box testing* merupakan pengujian pada tingkat alur perangkat lunak. Untuk mengetahui aplikasi *Cash Flow* tidak memiliki galat maka akan dilakukan uji coba dengan teknik *basis path* yang dimulai dari membuat *Flow Graph*, *Cyclomatic Complexity* (CC) dan melakukan *unit Case.* Penerapan dari metode pengujian *white box* menggunakan teknik *Basis path* pada aplikasi *Cash Flow* dapat mengevaluasi kompleksitas alur program, pada saat melakukan *unit test* dapat menentukan jumlah skenario pengujian yang akan dilakukan. Setelah dilakukan uji pada aplikasi *Cash Flow* dan tidak ditemukan galat, maka semua tes berhasil. Hasil pengujian ini dapat dijadikan acuan untuk memperbaiki aplikasi sesuai yang diharapkan, dan menjamin bahwa aplikasi yang dibuat sudah sesuai persyaratan.

Kata Kunci: White Box; Pengujian Perangkat Lunak; Android; Junit; Basis Path

1 Pendahuluan

Pada proses pengembangan sebuah sistem atau aplikasi terdapat beberapa tahapan, di antaranya proses analisa, perancangan, implementasi, uji coba, dan pengelolaan. Dari ke lima proses ini, proses uji coba adalah proses yang memerlukan waktu yang cukup lama. Untuk menjamin kualitas sebuah sistem atau aplikasi harus melalui tahapan uji coba. Ada dua metode yang digunakan dalam pengujian, yaitu secara fungsional (*Black Box*) dan secara sistematik (*White Box*) (Irawan, 2017). Aplikasi

yang memiliki kualitas buruk dan memiliki cacat dapat mengakibatkan biaya pengembangan meningkat, serta menambah waktu pengembangan aplikasi. Pengujian perangkat lunak memiliki fungsi penting dalam proses pengembangan aplikasi, dengan melakukan pengujian akan didapat seperti galat atau *error*, untuk mengetahui kualitas perangkat lunak tersebut apakah sudah memenuhi persyaratan kinerja yang baik dan menentukan perbedaan dari hasil yang diharapkan dengan hasil sebenarnya (Pratama, Ristianto, Prayogo, Nasrullah, & Saifudin, 2020).

Aplikasi yang akan diuji adalah *Cash Flow*, aplikasi ini berjalan pada sistem operasi android, dan memiliki fungsi untuk mendata alir kas seperti pemasukan dan pengeluaran kas dalam periode waktu tertentu. Aplikasi ini dibuat dengan sederhana dengan tujuan mempermudah penggunaan dan membuat perencanaan keuangan. Dalam kehidupan sehari-hari perencanaan keuangan memiliki peran penting dalam stabilitas keuangan. Salah satu fitur *Cash Flow* adalah mendata pengeluaran kas, pengujian perangkat lunak sangat penting dilakukan karena setiap orang membuat kesalahan pada saat pembuatan perangkat lunak, kesalahan pada masing – masing perangkat lunak akan berbeda (Ningrum, Suherman, Aryanti, Prasetya, & Saifudin, 2019). Jika terdapat galat atau *error* maka dapat menimbulkan salah penyampaian informasi, dan pengguna tidak mengetahui pengeluaran kas yang sudah terjadi. Penyampaian informasi yang salah juga dapat merusak rencana keuangan yang telah disusun oleh pengguna. Hal ini dapat dihindari dengan melakukan uji coba pada aplikasi *Cash Flow*. Metode yang akan digunakan dalam pengujian adalah *white box* dengan teknik unit test, pengujian perangkat lunak dilakukan dengan cek pada modul untuk dapat meneliti dan menganalisa kode dari program yang dibuat benar atau salah (Fatimah & Samsudin, 2019). Teknik yang digunakan adalah jalur dasar (*Basis Path*) untuk mengetahui kompleksitas logika (Herlambang, Rachmadi, Utami, Hakim, & Rohmah, 2019).

2 Metodologi

Berdasarkan pembahasan di atas metode yang digunakan adalah metode *white box* dan Teknik yang digunakan adalah teknik jalur dasar atau *basis path. white box testing* adalah pengujian perangkat lunak pada tingkat alur kode program, apakah masukan dan keluaran yang sesuai dengan spesifikasi yang dibutuhkan. (Cholifah, Yulianingsih, & Sagita, 2018), dan pengujian yang didasarkan pada pengujian *design* program secara prosedural, secara *strctural*, pengujian berbasis logika atau pengujian berbasis kode (Irawan, 2017). Metode jalur dasar adalah salah satu metode *white box testing*, di mana dalam proses pengujian diperlukan untuk membuat *flow graph* dari program skrip dan juga menentukan nilai kompleksitas siklomatik. Tes ini bertujuan untuk menganalisis kebenaran struktur program yang dibuat dan kinerja program (Rahayuda & Santiari, 2017). *Basis path* adalah suatu jalur unik yang melintasi alur progam dan tidak diperbolehkan terjadinya perulangan lintasan yang sama. Pada metode pengujian *basic path* mengharuskan menghitung kompleksitas logis dari alur program dan menggunakan ukuran sebagai petunjuk untuk mendefinisikan jumlah jalur eksekusi.

Dalam *white box testing* menggunakan *basis path* terdapat beberapa tahapan yaitu dengan membuat *flow graph* dari fungsi yang akan diuji, menghitung *cyclometic complexity* (CC) dan melakukan unittest (Sakethi, Kurniawan, & Tantriawan, 2014).

3 Pembahasan

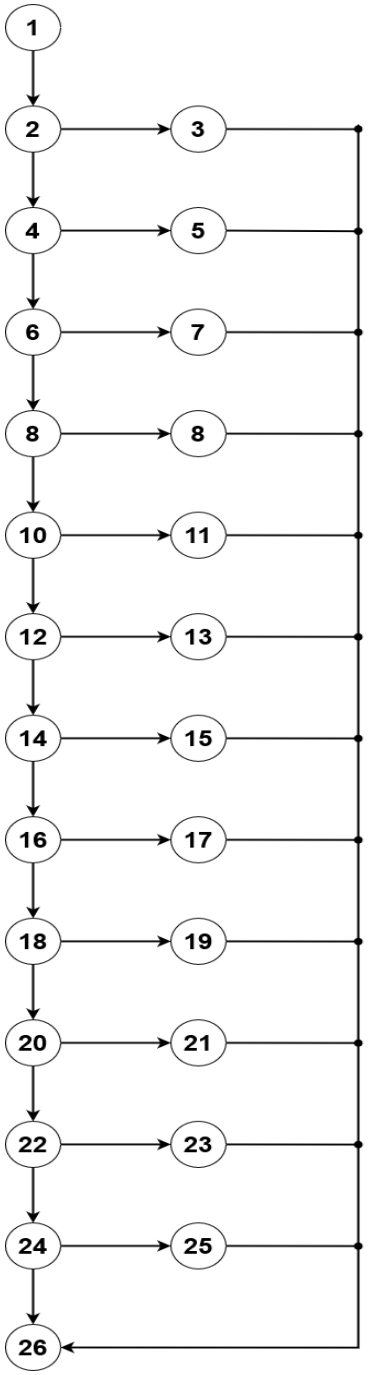
Pengujian di mulai dengan menghitung jumlah skenario yang akan di uji dengan menggunakan *cyclomatic complexity* (CC) dengan rumus v = e – n + 2 di mana e = jumlah jalur, dan n adalah jumlah simpul, untuk mengetahui nilai e dan n digunakan *flow graph*. Setelah mendapatkan nilai CC maka dibuat skenario uji. Pengujian menggunakan unit test, dan *Integrated Development Environment* (IDE) android studio. Berikut merupakan hasil hitung CC dan cuplikan kode pengujian:

1) Fungsi Bulan

   Fungsi bulan, adalah fungsi untuk mendapatkan bulan pada saat ini, dengan parameter berupa angka, berikut merupakan cuplikan kodenya:

```
private String getMonthString (String month) {

      switch (month) {
         case "01":
            return "JANUARY"

         case "02":
            return "FEBRUARY";

         case "03":
            return "MARCH";

         case "04":
            return "APRIL";

         case "05":
            return "MAY";

         case "06":
            return "JUNE";

         case "07":
            return "JULY";

         case "08":
            return "AUGUST";

         case "09":
            return "SEPTEMBER";

         case "10":
            return "OCTOBER";

         case "11":
           return "NOVEMBER";

         case "12":
           return "DECEMBER";

         default:
           return "";
      }
 }
```

Untuk menganalisa jalur eksekusi, selanjutnya dibuat *flow graph*. Berdasarkan *source code* (kode sumber) di atas maka dibuat *flow graph* seperti pada Gambar 1.



Gambar 1 *Flow graph* fungsi bulan

Berdasarkan flow graph pada Gambar 1, jumlah edge € = 37, jumlah node (n) = 26, sehingga nilai CC = 37 – 26 + 2 = 13.

Jalur independent yang didapat:

- Jalur 1 : 1-2-3-26
- Jalur 2 : 1-2-4-5-26
- Jalur 3 : 1-2-4-6-7-26
- Jalur 4 : 1-2-4-6-8-9-26
- Jalur 5 : 1-2-4-6-8-10-11-26

- Jalur 6 : 1-2-4-6-8-10-12-13-26
- Jalur 7 : 1-2-4-6-8-10-12-14-15-26
- Jalur 8 : 1-2-4-6-8-10-12-14-16-17-26
- Jalur 9 : 1-2-4-6-8-10-12-14-16-18-19-26
- Jalur 10 : 1-2-4-6-8-10-12-14-16-18-20-21-26
- Jalur 11 : 1-2-4-6-8-10-12-14-16-18-20-22-23-26
- Jalur 12 : 1-2-4-6-8-10-12-14-16-18-20-22-24-25-26
- Jalur 13 : 1-2-4-6-8-10-12-14-16-18-20-22-24-26

Setelah menghitung CC maka jumlah test yang dilakukan sebanyak 13 kali, berikut merupakan cuplikan kode yang digunakan dalam unit test:

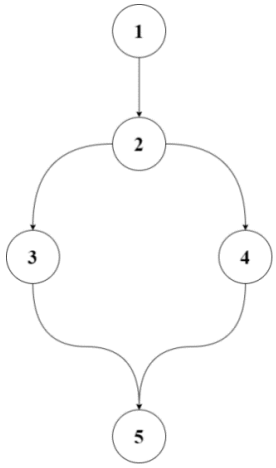
```
package com.fish.cashflow;

import android.support.test.runner.AndroidJUnit4;
import org.junit.Test;
import org.junit.runner.RunWith;
import static org.junit.Assert.assertEquals;

@RunWith(AndroidJUnit4.class)
public class GetMonthTest {

    @Test
    public void jan(){
        String tanggal = "01";
        String hasilYangDiHarapkan =
"JANUARY";
        String hasilKalkulasi =Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
    }

    @Test
    public void feb(){
        String tanggal = "02";
        String hasilYangDiHarapkan =
"FEBRUARY";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

    @Test
    public void mar(){
        String tanggal = "03";
        String hasilYangDiHarapkan = "MARCH";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

   @Test
     public void apr(){
        String tanggal = "04";
        String hasilYangDiHarapkan = "APRIL";
        String hasilKalkulasi = Helper
  .getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
     }

    @Test
    public void mei(){
        String tanggal = "05";
        String hasilYangDiHarapkan = "MAY";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

    @Test
    public void jun(){
        String tanggal = "06";
        String hasilYangDiHarapkan = "JUNE";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

    @Test
    public void jul(){
        String tanggal = "07";
        String hasilYangDiHarapkan = "JULY";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

    @Test
    public void agust(){
        String tanggal = "08";
        String hasilYangDiHarapkan =
"AUGUST";
        String hasilKalkulasi = Helper
.getMonthString(tanggal);
        assertEquals(hasilYangDiHarapkan,
                          hasilKalkulasi);
    }

    @Test
    public void sept(){
        String tanggal = "09";
        String hasilYangDiHarapkan =
```

```
"SEPTEMBER";
      String hasilKalkulasi = Helper
.getMonthString(tanggal);
      assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
   }

   @Test
   public void okt(){
      String tanggal = "10";
      String hasilYangDiHarapkan =
"OCTOBER";
      String hasilKalkulasi =  Helper
.getMonthString(tanggal);
      assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
   }

   @Test
   public void nov(){
      String tanggal = "11";
      String hasilYangDiHarapkan =
"NOVEMBER";
      String hasilKalkulasi = Helper
.getMonthString(tanggal);
      assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
   }

   @Test
   public void des(){
      String tanggal = "12";
      String hasilYangDiHarapkan =
"DECEMBER";
      String hasilKalkulasi = Helper
.getMonthString(tanggal);
      assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
   }

   @Test
   public void def(){
      String tanggal = "-1";
      String hasilYangDiHarapkan = "";
      String hasilKalkulasi = Helper
.getMonthString(tanggal);
      assertEquals(hasilYangDiHarapkan,
                        hasilKalkulasi);
   }
}
```

2) Fungsi Menambahkan Pengeluaran
   Fungsi ini digunakan untuk menerima masukan pengeluaran kas dari pengguna aplikasi dan data akan disimpan ke dalam basis data pada gawai ponsel. Berikut cuplikan kode:

```
public boolean insertDataExpense(int id,String
expense, String description, String date, String
category){

    SQLiteDatabase db =this
    .getWritableDatabase();
      ContentValues contentValues =
    new ContentValues();

      contentValues.put(COL_1_E, id);
      contentValues.put(COL_2_E, expense);
      contentValues.put(COL_3_E, description);
      contentValues.put(COL_4_E, date);
      contentValues.put(COL_5_E, category);

      long res =
    db.insert(TABLE_NAME_EXPENSE,
     null,contentValues);

        if(res == -1)
    return false;
        else
    return  true;
}
```

Untuk mengidentifikasi jalur eksekusi program, selanjutnya dibuat *flow graph*. Dari source code (kode sumber) di atas maka dibuat *flow graph* pada Gambar 2.



Gambar 2 *Flow graph* input pengeluaran

e= 5, n = 5 sehingga nilai CC = 5 – 5 + 2 = 2.
Jalur independent yang didapat:
- Jalur 1 : 1-2-3-5
- Jalur 2 : 1-2-4-5

Fungsi menambahkan pengeluaran memiliki nilai CC = 2, maka uji coba yang akan

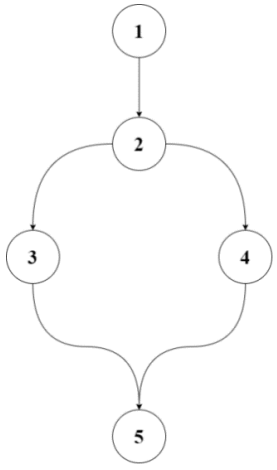
dilakukan sebanyak 2 skenario. Berikut cuplikan code:

```
package com.fish.cashflow;

import android.content.Context;
import android.support.test.InstrumentationRegistry;
import android.support.test.runner.AndroidJUnit4;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;


@RunWith(AndroidJUnit4.class)
public class DatabaseTest {

   private Context appContext;
   private DatabaseHelper db;

   private int id = 11;
   private String expanse = "50000";
   private String desc = "Test Save Expanses";
   private String date = "20180201";
   private String category =
"ENTERTAINMENT";

   @Before
   public void setup(){
      appContext = InstrumentationRegistry
.getTargetContext();
      db = new DatabaseHelper(appContext);
   }

   @Test()
   public void saveExpansesData(){
      boolean status = db.insertDataExpense(id,
expanse,desc,date,
category);
      assertTrue(status);
   }

@Test()
   public void saveExpansesData2(){
      boolean status2 = db.insertDataExpense(id,
expanse,desc,date,
category);
      assertFalse(status2);
   }
}
```

3) Fungsi Menambahkan Kategori
   Fungsi ini digunakan untuk menambah kategori pengeluaran dari pengguna aplikasi dan data akan disimpan kedalam basis data pada gawai ponsel, berikut cuplikan kode:

```
public boolean insertDataCategory(int id,String
            description, String budget, String state){

   SQLiteDatabase db = this.getWritableDatabase();
   ContentValues contentValues =
   new ContentValues();

   contentValues.put(COL_1_C, id);
   contentValues.put(COL_2_C, description);
   contentValues.put(COL_3_C, budget);
   contentValues.put(COL_4_C, state);

   long res
   =db.insert(TABLE_NAME_CATEGORY,
   null, contentValues);

      if(res == -1)
         return false;
      else
         return true;
    }
```

Dari cuplikan kode di atas maka dibuat *flow graph* sebagai berikut:



Gambar 3 Flow graph menambahkan kategori

e = 5, n = 5 sehingga nilai CC = 5 – 5 + 2 = 2.
Jalur independent yang didapat:

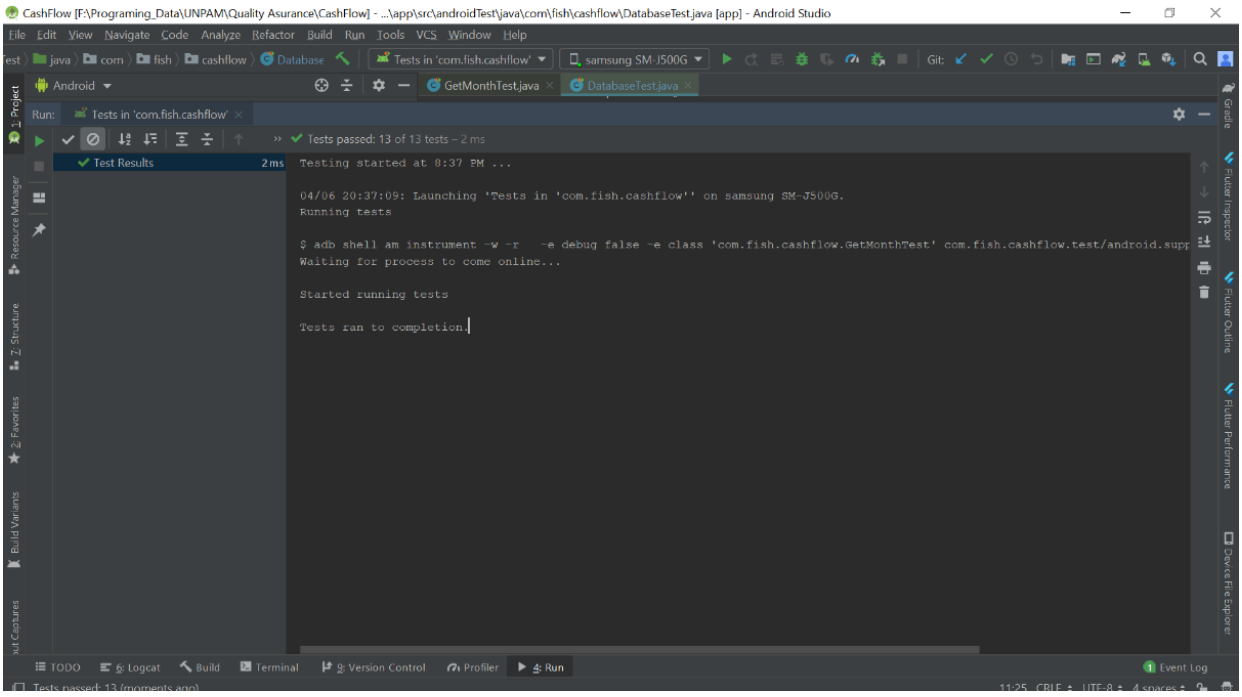
- Jalur 1 : 1-2-3-5
- Jalur 2 : 1-2-4-5

Fungsi menambahkan pengeluaran memiliki nilai CC = 2, maka uji coba yang akan dilakukan sebanyak 2 skenario. Berikut cuplikan code:

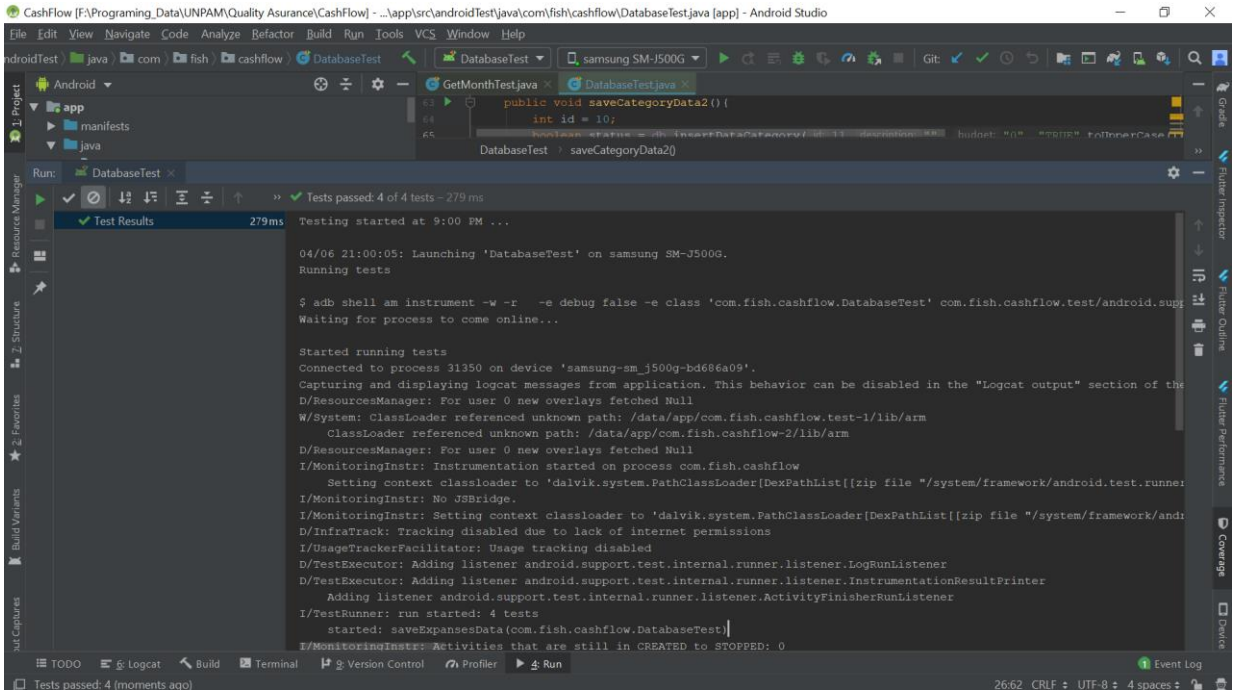
```
package com.fish.cashflow;

import android.content.Context;
import android.support.test.InstrumentationRegistry;
import android.support.test.runner.AndroidJUnit4;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

@RunWith(AndroidJUnit4.class)
public class DatabaseTest {

    private Context appContext;
    private DatabaseHelper db;
    private int id = 8;

    @Before
    public void setup(){
        appContext = InstrumentationRegistry
.getTargetContext();
        db = new DatabaseHelper(appContext);
    }

@Test()
    public void saveCategoryData(){
        boolean status =db.insertDataCategory(
id,"kategori","0",
"TRUE".toUpperCase());
        assertTrue(status);
    }

    @Test()
    public void saveCategoryData2(){
        boolean status = db.insertDataCategory(id,
"kategori", "0",
"TRUE".toUpperCase());
        assertFalse(status);
    }
}
```

4 Analisa dan Hasil Uji

Uji coba di awali dengan membuat *Flow Graph*, menghitung *Cyclometic Complexity* (CC), setelah mendapakan nilai kompleksitas alur program makan dilakukan *Unit Test.* Ada sebanyak 17 jalur yang uji pada aplikasi Cash Flow, 13 jalur pada fungsi mendapatkan kata bulan, 2 jalur pada fungsi memasukan data pengeluaran ke basis data, dan 2 fungsi memasukan data kategori pengeluaran ke basis data, berikut ini adalah hasil dari uji:



Gambar 4 Hasil uji fungsi bulan



Gambar 5 Hasil uji fungsi simpan basis data

Berdasarkan hasil uji yang didapat maka fungsi untuk memasukan data pengeluaran kas, memasukan data kategori pengeluaran kas dan pada fungsi untuk mendapatkan bulan berhasil tidak terjadi galat.

5 Kesimpulan

Metode pengujian *white box* menggunakan Teknik *Basis path* pada aplikasi cash flow dapat mengevaluasi kompleksitas alur program, pada saat melakukan *unit test* dapat menentukan jumlah skenario pengujian yang akan dilakukan. Pengujian kali ini dititikberatkan pada pengujian fungsi menyimpan, mengubah, dan menampilkan data di basis data. Semua fungsi diuji coba dan hasilnya berjalan sesuai harapan, akan tetapi ada di beberapa alur program fungsi tidak dibuat sebagaimana seharusnya dan tidak ada logika untuk menunjukkan apakah data berhasil disimpan pada basis data atau tidak.

6 Saran

Pengujian aplikasi *Cash Flow* ini diharapkan menjadi referensi untuk pengujian selanjutnya. Diharapkan pada pengujian selanjutnya menggunakan beberapa teknik pengujian Black Box untuk melangkapi dan dapat meningkatkan jaminan kualitas aplikasi.

Referensi

Cholifah, W. N., Yulianingsih, & Sagita, S. M. (2018). Pengujian Black Box Testing Pada Aplikasi Action & Strategy Berbasis Android Dengan Teknologi Phonegap. *Jurnal String*, 206-210.

Fatimah, & Samsudin. (2019). Perancangan Sistem Infromasi E-Jurnal pada Prodi Sistem. 33-49.

Handy, & Susilo, J. (2015). Jurnal Informatika dan Bisnis. *Aplikasi Pengujian White-Box Ibii Online Judge*, 56 - 68.

Herlambang, A. D., Rachmadi, A., Utami, K., Hakim, R. I., & Rohmah, N. (2019). Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK). *Pengembangan Fitur E-Matur Dengan V-Model Sebagai Alat Pengaduan Publik Untuk Website Badan Kepegawaian Negara*, 467 - 474.

Irawan, Y. (2017). Pengujian Sistem Informasi Pengelolaan Pelatihan Kerja UPT BLK Kabupaten Kudus. *Sentra Penelitian Engineering dan Edukasi*, 56-63.

Ningrum, F. C., Suherman, D., Aryanti, S., Prasetya, H. A., & Saifudin, A. (2019). Jurnal Informatika Universitas Pamulang. *Pengujian Black Box pada Aplikasi Sistem Seleksi Sales Terbaik Menggunakan Teknik Equivalence Partitions* , 125 - 130.

Pratama, B. P., Ristianto, I. B., Prayogo, I. A., Nasrullah, & Saifudin, A. (2020). Pengujian Perangkat Lunak Sistem Informasi Penilaian Mahasiswa dengan Teknik Boundary Value Analysis Menggunakan Metode Black Box

Testing. *Journal Of Artificial Intelligence And Innovative Applications*, 32-36.

Rahayuda, I. S., & Santiari, N. L. (2017). Jurnal Ilmiah Kursor Menuju Solusi Teknologi Informasi. *Basis Path Testing Of Iterative Deepening Search And Held-Karp On Pathfinding Algorithm*, 39 - 48.

Sakethi, D., Kurniawan, D., & Tantriawan, H. (2014). Pengujian dan Perawatan Sistem Informasi Menggunakan White Box Testing. *Ilmu Komputer Unila Publishing Network all right reserved*, 27-35.

Utomo, D. W., Kurniawan, D., & Astuti, Y. P. (2018). Teknik Pengujian Perangkat Lunak Dalam Evaluasi Sistem Layanan Mandiri Pemantauan Haji Pada Kementerian Agama Provinsi Jawa Tengah. *Jurnal SIMETRIS* , 731-756.

Nidhra, S., & Dondeti, J. (2012). Black Box And White Box Testing Techniques –A Literature Review. *International Journal of Embedded Systems and Applications (IJESA)* , 29-50.

Syaikhuddin, M. M., Anam, C., Rinaldi, A. R., & Conoras, M. E. (2018). Conventional Software Testing Using White Box Method. *KINETIK* , 67-74.