



**TC
FIRAT ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**

YMH459 YAZILIM MÜHENDİSLİĞİ GÜNCEL KONULAR DERSİ PROJESİ
Test Dokümanı

Proje Adı

CRYPIT(Fotoğraf Şifreleme için Mobil Uygulama)

Proje Logosu



Proje Ekip Lideri

SÜMEYYE GÜLNUR DADAK (16542503)

Proje Çalışma Grubu

ABDULKADİR DOĞANER (16541563)
ARZU KÜBRA YILAR (175541031)
BERNA UZUNOĞLU (16542513)
ENİS CAN YILMAZ (15541531)
HÜSEYİN BİTİKÇİ (16541509)
İHSAN CENKİZ (175541022)
İSMAİL ÇAĞAN (16541542)
KÜBRA ATICI (14545522)
MEHMET CAN AKKAŞ (14542518)
MEHMET FURKAN KEMALLI (185541087)
MERT BEKTAŞ (16541522)
MUSTAFA ENES ÖZÇELİK (175541068)
TUGAY AYAR (15542514)

Proje Yürütücüleri

Doç. Dr. Fatih ÖZKAYNAK

**ELAZIĞ
2020-2021**

İçindekiler Tablosu

| | |
|--|----|
| 1.Giriş | 4 |
| 1.1 Yazılım Test Ekibi Kimdir | 4 |
| 1.2 Test ekibi ve yaptıkları testler: | 4 |
| 1.3 Birim test: | 4 |
| 1.4 Performans test: | 4 |
| 1.5 Mobil Uygulama test: | 4 |
| 1.6 Beyaz Kutu test: | 4 |
| 1.7 Yazılım Kalite Metrikleri: | 4 |
| 2.BİRİM TESTİ | 5 |
| 3.PERFORMANS TESTİ | 11 |
| 4.BEYAZ KUTU TESTİ | 15 |
| 4.1 YAZILIM MÜHENDİSLİĞİ GÜNCEL KONULAR PROJESİ BEYAZ KUTU TEST RAPORLARI; | 15 |
| 4.1.1 Zigzagfunction(); | 16 |
| 4.1.2 DecimaltıBinalyFunction(); | 17 |
| 4.1.3 DivisionHexFunction(); | 18 |
| 4.1.4 xorFunction(); | 19 |
| 4.1.5 byteConversionFunction(); | 20 |
| 4.1.6 ReversezigzagFunction(); | 21 |
| 4.1.7 İmageToFirestore(); | 22 |
| 4.1.8 PortİslemiYap(); | 23 |
| 4.1.9 List createSenderKey(); | 24 |
| 4.1.10 List randArr(); | 25 |
| 4.1.11 List randomBin(); | 26 |
| 4.1.12 List createReceiverKey(); | 27 |
| 4.1.13 int wp1(); | 28 |
| 5.KALİTE METRİKLERİ | 30 |
| 5.1 LCOM | 30 |
| 5.2 DÖNGÜSEL KARMAŞIKLIK | 33 |
| 5.2.1 Zigzagfunction(); | 33 |
| 5.2.2 DivisionHexFunction(); | 34 |
| 5.2.3 xorFunction(); | 34 |
| 5.2.4 byteConversionFunction(); | 34 |
| 5.2.5 ReversezigzagFunction(); | 34 |

| | |
|----------------------------------|----|
| 5.2.6 imageToFirestore(); | 34 |
| 5.2.7 PortIslemiYap(); | 34 |
| 5.2.8 createSenderKey(); | 34 |
| 5.2.9 randArr(); | 35 |
| 5.2.10 randomBin(); | 35 |
| 5.2.11 createReceiverKey(); | 35 |
| 5.2.12 wp1(); | 35 |
| 5.3 SLOC ve LSLOC | 35 |
| 5.2.1 Zigzagfunction(); | 35 |
| 5.2.2 DecimaltıBinalyFunction(); | 35 |
| 5.2.3 DivisionHexFunction(); | 35 |
| 5.2.4 xorFunction(); | 36 |
| 5.2.5 byteConversionFunction(); | 36 |
| 5.2.6 ReversezigzagFunction(); | 36 |
| 5.2.7 imageToFirestore(); | 36 |
| 5.2.8 PortIslemiYap(); | 36 |
| 5.2.9 createSenderKey(); | 36 |
| 5.2.10 randArr(); | 36 |
| 5.2.11 randomBin(); | 36 |
| 5.2.12 createReceiverKey(); | 36 |
| 5.2.13 wp1(); | 36 |
| 6. MOBİL TESTİ | 37 |
| 6.1 Fonksiyonel Testi: | 38 |
| 6.2 Kullanılabilirlik Testi: | 39 |
| 6.3 Kullanıcı Arayüz Testi: | 40 |
| 6.4 Uygunluk Testi: | 40 |
| 6.5 Performans Testi: | 41 |
| 6.6 Güvenlik testi: | 41 |
| 6.7 Kurtarma Testi: | 42 |
| 6.8 Yerelleştirme testi: | 42 |
| 6.9 Değişiklik testi: | 42 |
| 6.10 Beta Testi: | 43 |
| 6.11 Sertifikasyon Testi: | 43 |

YAZILIM TEST DÖKÜMANI

1.Giriş

1.1 Yazılım Test Ekibi Kimdir

Yazılım test ekibi, oyun sistemleri veya mobil uygulamalar gibi yeni yazılım ürünlerinin problemlerini tespit etmekle sorumludur. Yazılım üzerinde çeşitli testler yapar. Sorunları tanımlar ve gerektiğinde hata ayıklama programları çalıştırır.

Yukarıda test ekibinin görevi tanımlanmıştır burada ise projede çalışan test ekibini ve uyguladıkları testleri göreceğiz.

1.2 Test ekibi ve yaptıkları testler:

| | |
|---------------------|--|
| İsmail ÇAĞAN | Birim test, performans testi ve mobil test |
| Tugay AYAR | Birim test, performans testi ve mobil test |
| Abdul Kadir DOĞANER | Katılım Sağladı |
| Arzu YILAR | Beyaz kutu, Kalite metrikleri |
| Kübra ATICI | Beyaz kutu, Kalite metrikleri |

1.3 Birim test: Programcının yazdığı fonksiyonu istenilen sonucu göre doğru olup olmadığını test etmek için kullanır. Diğer bir anlamla Bu yöntemde yazılımcı yazılım kodunu oluşturan birimlerin kullanıma hazır olduğuna ikna olur.

1.4 Performans test: Performans testi, sistemin belirli durumlarda, belirlenen beklentileri verip vermediğini kontrol etmek amacıyla yapılan testlerdir. Amacı sistemin belirli bir yük altındaki performansının ölçülmesi ve istenilen performansa ulaşmasını sağlamaktır.

1.5 Mobil Uygulama test: Mobil uygulamaların ara yüzlerinin ve kodlarının farklı aşamalardan geçirilerek test edilmesidir. Mobil uygulamaların test adımları çok sayıda farklı testlerin birleştirilmesinden oluşmaktadır.

Mobil uygulama testini uygulama stratejinin bir parçası olarak benimseyerek, uygulama deneyiminizi sürekli olarak aşmak için veri destekli, eyleme geçirilebilir öngörülere güvenebilir ve sonuç olarak kullanıcı katılımı, elde tutma ve para kazanma dahil olmak üzere uygulama temel metriklerini artırabilirsiniz. Uygulamanın her bir ögesini, akışını ve özelliğini optimize etmek için test öğelerini kullanmak, her kullanıcı temas noktasında dönüşümleri en üst düzeye çıkarmaya ve nihayetinde tüm kullanıcı yolculuğunu kolaylaştırmaya yardımcı olabilir.

1.6 Beyaz Kutu test: Literatürde beyaz kutu, cam kutu, saydam kutu gibi anlamlarla ifade edilen white box testi, sistemin iç yapısını bilen kişiler tarafından yapılan yazılım testine verilen tekniktir. Beyaz kutu testi genellikle yazılım mühendisleri tarafından yapılır. Özellikle bir yazılım birimini hazırlayan geliştirici, birim testler(unit testing) hazırlayarak süreci başlatır.

1.7 Yazılım Kalite Metrikleri: Yazılım kalite metriği yazılımların bir çok yönden değerlendirilmesini sağlayan, kaliteli bir yazılım geliştirmede bize yardımcı olan elemanlardır. Bunlar çeşitli alt yapılara bölünürler her yapının temsil ettiği bir değer ve her değer bir anlamı vardır. Karmaşık yapılarda bu değerler üzerinden uygulama hakkında çeşitli yorumlar yapılabilir.

2.BİRİM TESTİ

```
List<int> array1=[1,2,3,4,5,6,7,8,9];
zigzagFunction(array) {
    var array2 = [];
    var array3 = [];
    var array4 = [];
    for (var i = 0; i < array.length; i++) {
        if (i % 2 == 0) {
            array2.insert(0, array[i]);
        } else {
            array2.add(array[i]);
        } }
    for (var i = 0; i < array.length; i++) {
        if (i % 5 == 0) {
            array3.insert(0, array2[i]);
        } else {
            array3.add(array2[i]);
        } }
    for (var i = 0; i < array.length; i++) {
        if (i % 3 == 0) {
            array4.insert(0, array3[i]);
        } else {
            array4.add(array3[i]);
        } }
    return array4;
}

void main()
{
    bool sonuc;
    var kod=[];
    var beklenensonuc=[4, 5, 2, 9, 7, 3, 1, 6, 8];

    kod=(zigzagFunction(array1));
    for(int i=0;i<beklenensonuc.length;i++)
    {
        if(kod[i]==beklenensonuc[i])
        { sonuc=true; }

        else { sonuc=false; }
    }
    print(sonuc);
}
```

▶ RUN

Console

true

Documentation

```

var array1= [1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0];
decimaltoBinary(array) {
  var binaryList = [];
  var lastArr = [];
  for (var i = 0; i < array.length; i++) {
    binaryList.add(int.parse(array[i].toString(), radix: 10).toRadixString(2));
  }
  for (var j = 0; j < binaryList.length; j++) {
    var tempArr = [binaryList[j]];
    for (var k = 1; k < tempArr[0].length; k++) {
      lastArr.add(tempArr[0][k]);
    }
  }
  return binaryList;
}
void main()
{
  bool sonuc;
  var kod=[];
  var beklenensonuc=[1, 10, 11, 100, 101, 110, 111, 1000, 1001];

  kod=decimaltoBinary(array1);
  print(kod);
  print(beklenensonuc);

  for(int i=0;i<beklenensonuc.length;i++)
  {
    if(beklenensonuc[i] == kod[i])
    {
      sonuc=true;
    }
    else
    {
      sonuc=false;
    }
  }
  print(sonuc);
}

```

▶ RUN

Console

```

[1, 10, 11, 100, 101, 110, 111, 1000, 1001]
[1, 10, 11, 100, 101, 110, 111, 1000, 1001]
true

```

Documentation

```

List<String> array1=["t","e","s","t","e","k","i","b","i"];
divisonHexFunction(List array) {
    var binaryList = array;
    var lastArr = [];

    for (var j = 0; j < binaryList.length; j++) {
        var a = binaryList[j].toString();
        var uzunluk = 8 - a.length;
        var tempArr = ['1' + '0' * uzunluk + a];
        for (var k = 1; k < tempArr[0].length; k++) {
            lastArr.add(tempArr[0][k].toString());
        }
    }
    return lastArr;
}

void main()
{
    bool sonuc;
    var kod=[];
    var beklenensonuc=["0", "0", "0", "0", "0", "0", "0", "0", "t", "0", "0",
"0", "0", "0", "0", "0", "0", "e", "0", "0", "0", "0", "0", "0", "0", "s", "0",
"0", "0", "0", "0", "0", "0", "t", "0", "0", "0", "0", "0", "0", "0", "e",
"0", "0", "0", "0", "0", "0", "0", "0", "k", "0", "0", "0", "0", "0", "0", "0",
"i", "0", "0", "0", "0", "0", "0", "0", "b", "0", "0", "0", "0", "0", "0", "0",
"0", "i"];

    kod=(divisonHexFunction(array1));
    for(int i=0;i<beklenensonuc.length;i++)
    {
        if(kod[i]==beklenensonuc[i])
        { sonuc=true; }

        else { sonuc=false; }
    }
    print(sonuc);
}

```

▶ RUN

Console

true

Documentation

```
List<String> array1=["0","1","1","1","0","1","0","1","0"];
List<int> array2=[1,0,1,0,1,0,1,0,1];

xorFunction(array1, array2) {
    var a = array1;
    var b = array2;
    var c = [];
    for (var i = 0; i < b.length; i++) {
        c.add(int.parse(a[i]) ^ b[i]);
    }
    return c;
}

void main()
{
    bool sonuc;
    var kod=[];
    var beklenensonuc=[1, 1, 0, 1, 1, 1, 1, 1, 1];
    kod=(xorFunction(array1,array2));
    for(int i=0;i<beklenensonuc.length;i++)
    {
        if(kod[i]==beklenensonuc[i])
        { sonuc=true; }

        else { sonuc=false; }
    }
    print(sonuc);
}
```

Console

true

Documentation

```
import 'dart:math';
List<int> array1=[1,0,1,0,1,0,1,0,1];

byteConversionFunction(array) {
    var c = array;
    var array1 = [];
    var z = '';
    for (var i = 0; i < c.length; i++) {
        z += c[i].toString();
        if (i % 8 == 7) {
            var sum = 0;
            for (var j = 0; j < z.length; j++) {
                if (z[j] == "1") {
                    var i = pow(2, 7 - j).toString(); //2 lik tabana çevirme
                    sum += int.parse(i);
                }
            }
            array1.add(sum);
            z = '';
        }
    }
    return array1;
}

void main()
{
    bool sonuc;
    var kod=[];
    var beklenensonuc=[170];
    kod=(byteConversionFunction(array1));
    for(int i=0;i<beklenensonuc.length;i++)
    {
        if(kod[i]==beklenensonuc[i])
        { sonuc=true; }

        else { sonuc=false; }
    }
    print(sonuc);
}
```

Console

true

Documentation

▶ RUN

```
List<int> array1=[1,2,3,4,5,6,7,8,9];

reversezigZagFunction(Iarray) {
    var I_BYTELIST = Iarray;
    var I_DEGISKENLIST1 = [];
    var I_DEGISKENLIST2 = [];
    var I_DEGISKENLIST3 = [];

    var STR_BYTE_LIST_LEN = ((I_BYTELIST.length - 1) / 3).toString();
    var I_BYTE_LIST_LEN = int.parse(STR_BYTE_LIST_LEN[0]);
    var I_DEGISKEN1 = 0;
    var I_DEGISKEN2 = 1;
    var I_DEGISKEN3 = 0;
    for (var i = 0; i < I_BYTE_LIST_LEN; i++) {
        I_DEGISKENLIST1.add(I_BYTELIST[i]);
    }
    for (var i = I_BYTE_LIST_LEN; i < I_BYTELIST.length; i++) {
        I_DEGISKENLIST2.add(I_BYTELIST[i]);
    }
    I_DEGISKENLIST3.add(I_DEGISKENLIST2[0]);

    I_DEGISKENLIST1 = I_DEGISKENLIST1.reversed.toList();
    while (true) {
        if (I_DEGISKEN1 == I_DEGISKENLIST2.length - 1) {
            break;
        }
        I_DEGISKENLIST3.add(I_DEGISKENLIST2[I_DEGISKEN2]);
        if ((I_DEGISKEN2 % 2 == 0) && (I_DEGISKEN3 < I_DEGISKENLIST1.length)) {
            I_DEGISKENLIST3.add(I_DEGISKENLIST1[I_DEGISKEN3]);
            I_DEGISKEN3++;
        }
        I_DEGISKEN2++;
        I_DEGISKEN1++;
    }
    I_BYTELIST = I_DEGISKENLIST3;
    I_DEGISKENLIST1 = [];
    I_DEGISKENLIST2 = [];
    var I_BYTE_LIST_LEN1 = 0;
    if ((I_BYTELIST.length % 5 == 1) || (I_BYTELIST.length % 5 == 2)) {
        I_BYTE_LIST_LEN1 = (I_BYTELIST.length / 5).round();
    } else {
        I_BYTE_LIST_LEN1 = (I_BYTELIST.length / 5 - 1).round();
    }
    for (var i = I_BYTE_LIST_LEN1; i >= 0; i--) {
```

Console

true

Documentation

Yukarıdaki fonksiyonlar beklentiler ve çıktıları ile aynı sonucu verdiği için doğru(TRUE) çalışmıştır ve birim testini başarı ile geçmiştir.

```

1  import 'dart:convert';
2  import 'package:hex/hex.dart';
3  import 'package:sha3/sha3.dart';
4  |
5  // İş Paketi 1 Ramden Veri Okuma, SHA-3 ten geçirerek int değere dönüştürme.
6  int wp1() {
7      var a = [].hashCode.toString();
8      var k = SHA3(256, KECCAK_PADDING, 256);
9      k.update(utf8.encode(a));
10     var hash = k.digest();
11     var myHex = (HEX.encode(hash)).toString();
12     var wp1OutArr = [];
13     var wp1Out = 0;
14     for (var i = 0; i <= myHex.length - 8; i += 8) {
15         final hex = myHex.substring(i, i + 8);
16         final number = int.parse(hex, radix: 16);
17         wp1OutArr.add(number);
18     }
19     for (var i = 0; i < wp1OutArr.length; i++) {
20         if (wp1OutArr[i] != null) {
21             wp1Out += wp1OutArr[i];
22         }
23     }
24     return wp1Out;
25 }
26 void main(){
27     print(wp1());
28 }

```

PROBLEMS 52

OUTPUT

DEBUG CONSOLE

TERMINAL

```

[Running] dart "c:\Users\Tugay\Desktop\LockerProject-main\test.dart"
17958489879

```

Bu fonksiyon her çalıştığında farklı bir değer üretecektir ve bunu başarı ile sağladığı için birim testi başarılıdır.

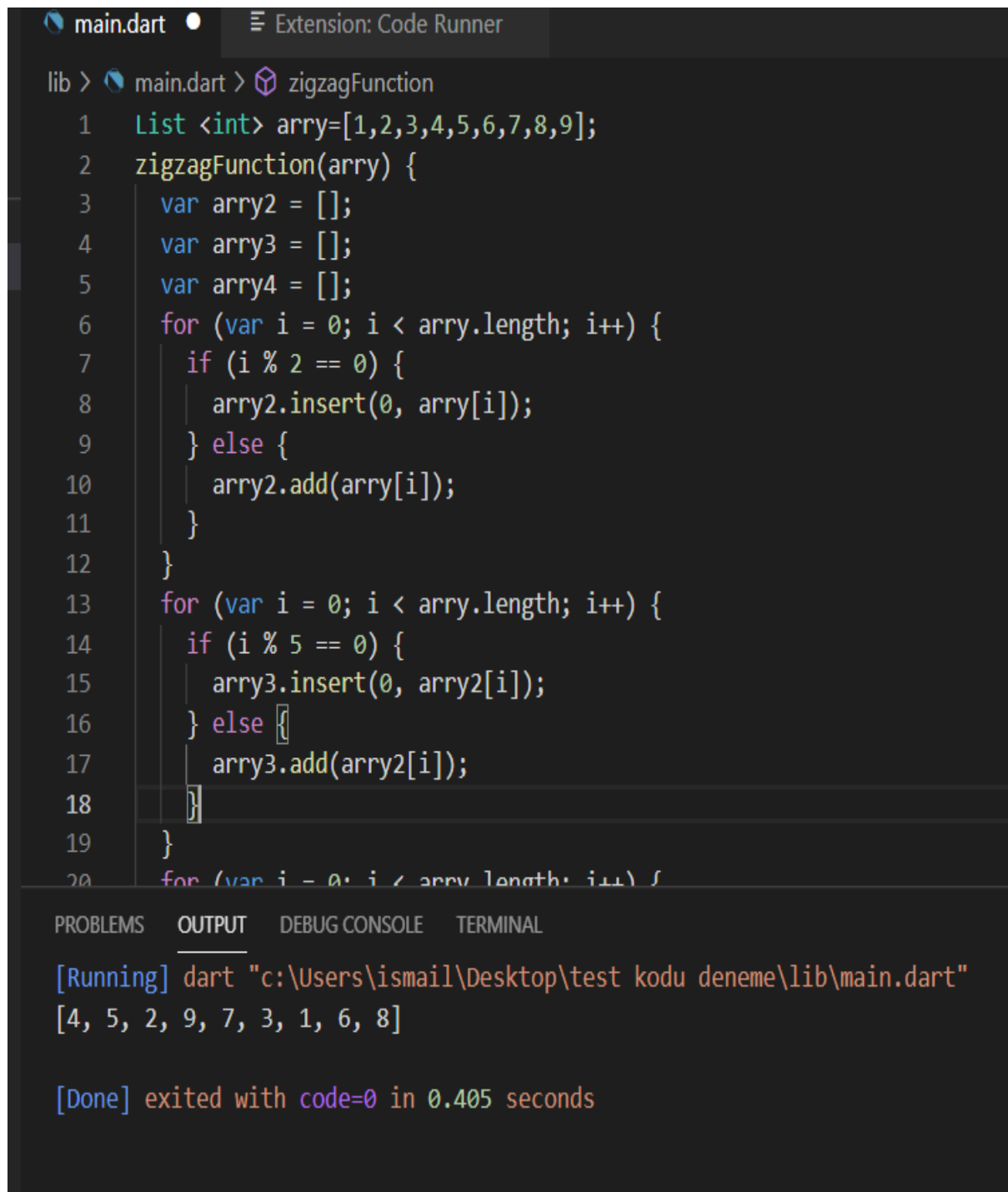
```

1 import 'dart:convert';
2 import 'package:hex/hex.dart';
3 import 'package:sha3/sha3.dart';
4 // Gönderici için anahtar üreten fonksiyon.
5
6 List createSenderKey(var bytes) {
7   // wp2Out ==> Hocanın verdiği değerlerdir. Değiştirilmemesi gerekir.
8   var wp2Out = [
9     0.868108929091533,
10    0.436522798444119,
11    0.850347542213392,
12    0.775550139264662,
13    0.752164288690584,
14    0.117648319430803,
15    0.838178032758474
16  ];
17   var randomArray = randArr(wp2Out, bytes);
18   return randomArray;
19 }
20
21 List randArr(var wp2Out, var bytes) {
22   var t = bytes * 8;
23   var tempRandArr = [];
24   var randArr = [];
25   // Kullanılan başlangıç değerlerinin indisleri.
26   var start_values_indices = [];
27   while (true) {
28     var start_value_index = (wp1() % 7);
29     var start_value = wp2Out[start_value_index];

```

Yukarıdaki fonksiyona bağlı olarak çalıştığı için herseferinde farklı anahtarlar üreteceğinden birim testi başarılıdır

3.PERFORMANS TESTİ



The image shows a code editor window with a file named `main.dart` and an extension named `Code Runner`. The code defines a `zigzagFunction` that takes an array of integers and returns a new array. The function uses two nested loops to process the input array. The first loop iterates over the array, and the second loop iterates over the result array. The output of the function is displayed in the `OUTPUT` tab.

```
lib > main.dart > zigzagFunction
1 List<int> array=[1,2,3,4,5,6,7,8,9];
2 zigzagFunction(array) {
3   var array2 = [];
4   var array3 = [];
5   var array4 = [];
6   for (var i = 0; i < array.length; i++) {
7     if (i % 2 == 0) {
8       array2.insert(0, array[i]);
9     } else {
10      array2.add(array[i]);
11    }
12  }
13  for (var i = 0; i < array.length; i++) {
14    if (i % 5 == 0) {
15      array3.insert(0, array2[i]);
16    } else {
17      array3.add(array2[i]);
18    }
19  }
20  for (var i = 0; i < array.length; i++) {
```

PROBLEMS **OUTPUT** **DEBUG CONSOLE** **TERMINAL**

[Running] dart "c:\Users\ismail\Desktop\test kodu deneme\lib\main.dart"

[4, 5, 2, 9, 7, 3, 1, 6, 8]

[Done] exited with code=0 in 0.405 seconds

```
main.dart • Extension: Code Runner
lib > main.dart > decimaltoBinaryFunction
1 List<int> array1=[1,2,3,4,5,6,7,8,9];
2 decimaltoBinaryFunction(array) {
3   var binaryList = [];
4   var lastArr = [];
5   for (var i = 0; i < array.length; i++) {
6     binaryList.add(int.parse(array[i].toString(), radix: 10).toRadixString(2));
7   }
8
9   for (var j = 0; j < binaryList.length; j++) {
10    var tempArr = [binaryList[j]];
11    for (var k = 1; k < tempArr[0].length; k++) {
12      lastArr.add(tempArr[0][k]);
13    }
14  }
15  return binaryList;
16 }
Run | Debug
17 void main()
18 {
19   print(decimaltoBinaryFunction(array1));
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[Running] dart "c:\Users\ismail\Desktop\test kodu deneme\lib\main.dart"

[1, 10, 11, 100, 101, 110, 111, 1000, 1001]

[Done] exited with code=0 in 0.582 seconds

```
main.dart • Extension: Code Runner
lib > main.dart > divisonHexFunction
1 List<String> array1=["t","e","s","t","e","k","i","b","i"];
2
3 divisonHexFunction(List array) {
4   var binaryList = array;
5   var lastArr = [];
6
7   for (var j = 0; j < binaryList.length; j++) {
8     var a = binaryList[j].toString();
9     var uzunluk = 8 - a.length;
10    var tempArr = ['1' + '0' * uzunluk + a];
11    for (var k = 1; k < tempArr[0].length; k++) {
12      lastArr.add(tempArr[0][k].toString());
13    }
14  }
15  return lastArr;
16 }
Run | Debug
17 void main()
18 {
19   print(divisonHexFunction(array1));
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

[Running] dart "c:\Users\ismail\Desktop\test kodu deneme\lib\main.dart"

[0, 0, 0, 0, 0, 0, 0, 0, t, 0, 0, 0, 0, 0, 0, 0, e, 0, 0, 0, 0, 0, 0, s, 0, 0, 0, 0, 0, 0, 0, i, 0, 0, 0, 0, 0, 0, 0, b, 0, 0, 0, 0, 0, 0, i]

[Done] exited with code=0 in 0.358 seconds

```
main.dart • Extension: Code Runner
lib > main.dart > ...
1 List <String> array1=["1","1","0","1","0","1","0","1","0"];
2 List <int> array2=[1,0,1,0,1,0,1,0,0];
3
4 xorFunction(array1, array2) {
5     var a = array1;
6     var b = array2;
7     var c = [];
8     for (var i = 0; i < b.length; i++) {
9         c.add(int.parse(a[i]) ^ b[i]);
10    }
11    return c;
12 }
Run | Debug
13 void main()
14 {
15     print(xorFunction(array1,array2));
16 }
17

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] dart "c:\Users\ismail\Desktop\test kodu deneme\lib\main.dart"
[0, 1, 1, 1, 1, 1, 1, 1, 0]

[Done] exited with code=0 in 0.372 seconds
```

```
main.dart • Extension: Code Runner
lib > main.dart > byteConversionFunction
1 import 'dart:math';
2 List <int> array1=[1,0,1,0,1,0,1,0,0];
3 byteConversionFunction(array) {
4     var c = array;
5     var array1 = [];
6     var z = '';
7     for (var i = 0; i < c.length; i++) {
8         z += c[i].toString();
9         if (i % 8 == 7) {
10             var sum = 0;
11             for (var j = 0; j < z.length; j++) {
12                 if (z[j] == "1") {
13                     var i = pow(2, 7 - j).toString(); //2 lik tabana çevirme
14                     sum += int.parse(i);
15                 }
16             }
17             array1.add(sum);
18             z = '';
19         }
20     }
21     return array1;
22 }
Run | Debug
19 void main()
20 {
21     print(byteConversionFunction(array1));
22 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] dart "c:\Users\ismail\Desktop\test kodu deneme\lib\main.dart"
[0, 1, 1, 1, 1, 1, 1, 1, 0]

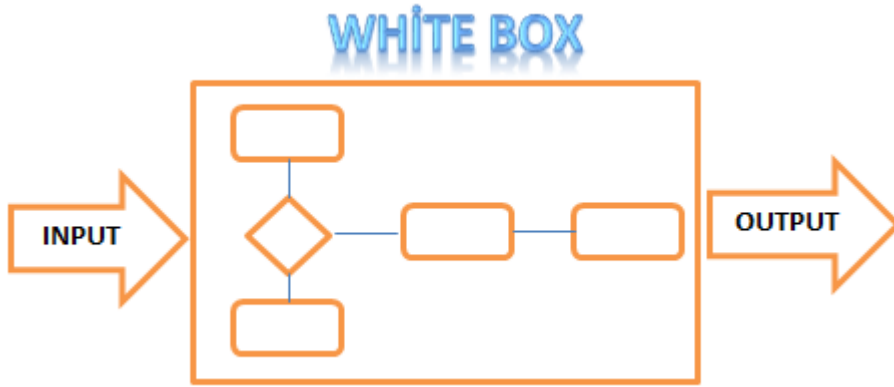
[Done] exited with code=0 in 0.365 seconds
```


4.BEYAZ KUTU TESTİ

Beyaz Kutu, Testleri Geliştirilen Yazılımın Kod Yapısı Bilinerek Gerçekleştirilen Test Tasarım Tekniğidir.

Beyaz Kutu testleri Geliştirilen Yazılımın İç Yapısı Ve İş Akışlarıyla ilgilenir.**Beyaz Kutu Test Tasarım** Tekniği Veri Akışlarına, Kontrol Akışlarına, İfade Kapsama, Dal Kapsama Gibi Konulara Odaklanır.

Beyaz Kutu Testleri **Birim, Tümlleştirme Ve Sistem Test Seviyelerinde** Gerçekleştirilebilir. Birim Test Seviyesinde Gerçekleştirilen Beyaz Kutu Testleri Birim Tümlleştirme Öncesinde Birimdeki Hataları Bulmayı Amaçlar. Tümlleştirme Seviyesindeki Beyaz Kutu Testleri İse Modüllerin Birbiri İle İletişiminde Ortaya Çıkabilecek Olan Hataları Bulmak Hedeflenir. Sistem Seviyesinde Gerçekleştirilen Beyaz Kutu Testlerinde İse Amaç Kapsama Analizlerinin Gerçekleştirilmesidir.

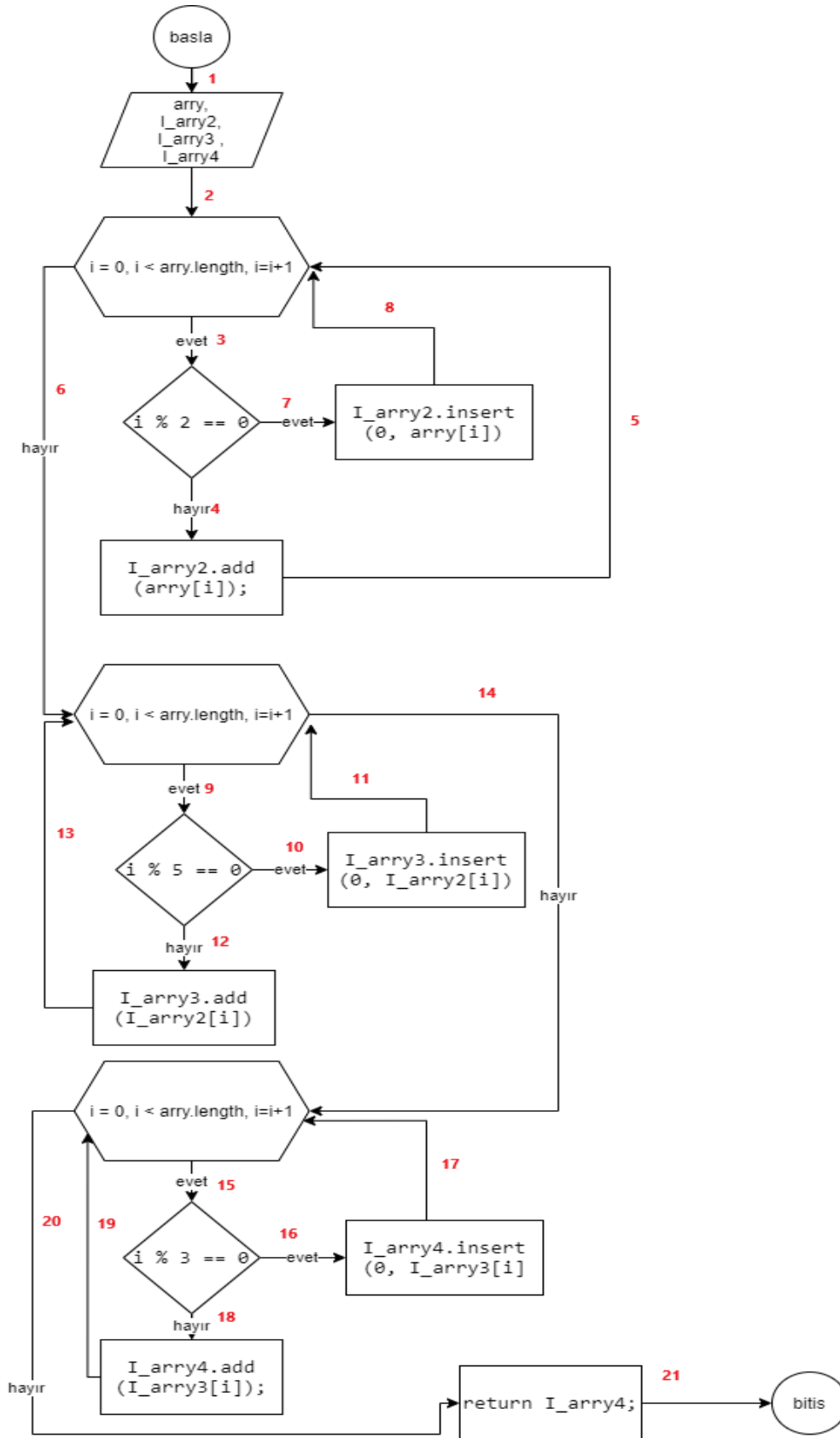


Şekil 1: Beyaz Kutu Testi

4.1 YAZILIM MÜHENDİSLİĞİ GÜNCEL KONULAR PROJESİ BEYAZ KUTU TEST RAPORLARI;

Yapılan projenin kodlarını fonksiyonlarına göre ayırıp beyaz kutu testi yapıldı.

4.1.1 Zigzagfunction();

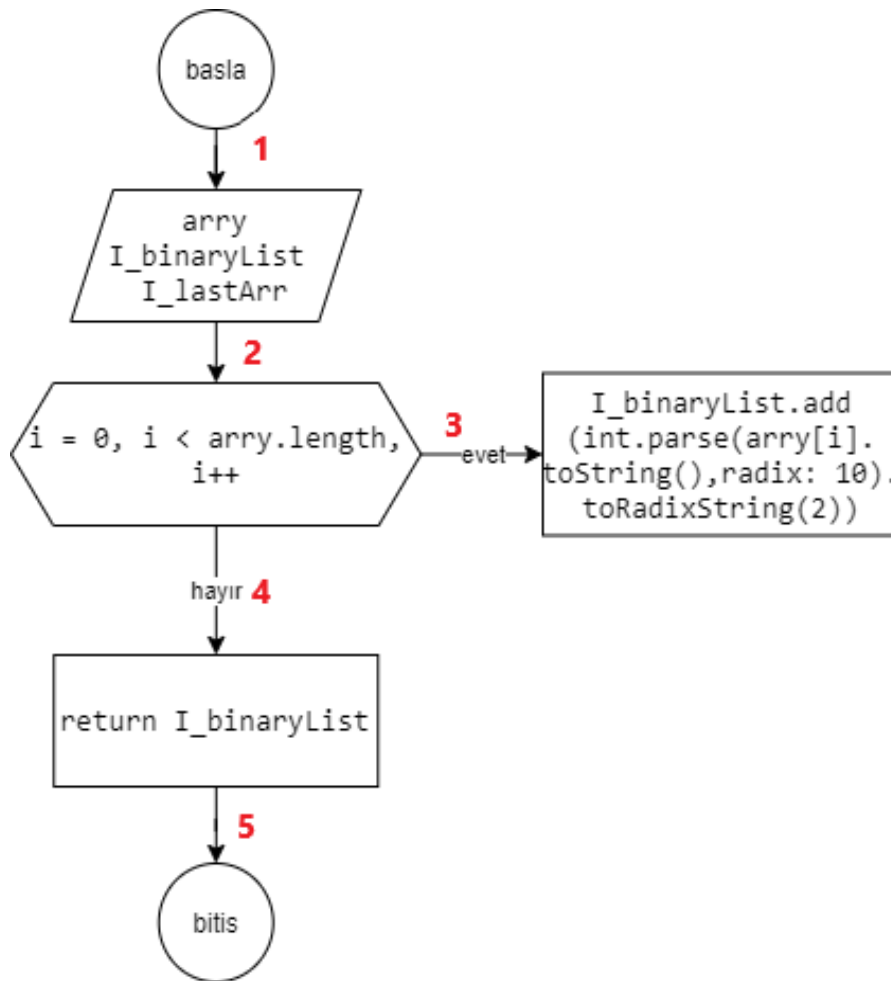


Şekil 2: ZigzagFunction fonksiyonu Akış Şeması

1-2-3-4-5
1-2-3-7-8
1-2-6-9-10-11
1-2-6-9-12-13
1-2-6-14-15-18-19
1-2-6-14-15-16-17
1-2-6-14-20-21

Sırasıyla yukarıda ki adımlar denendi ve zigzagfunction adımları başarıyla çalıştı.

4.1.2 DecimAltıBinalyFunction();



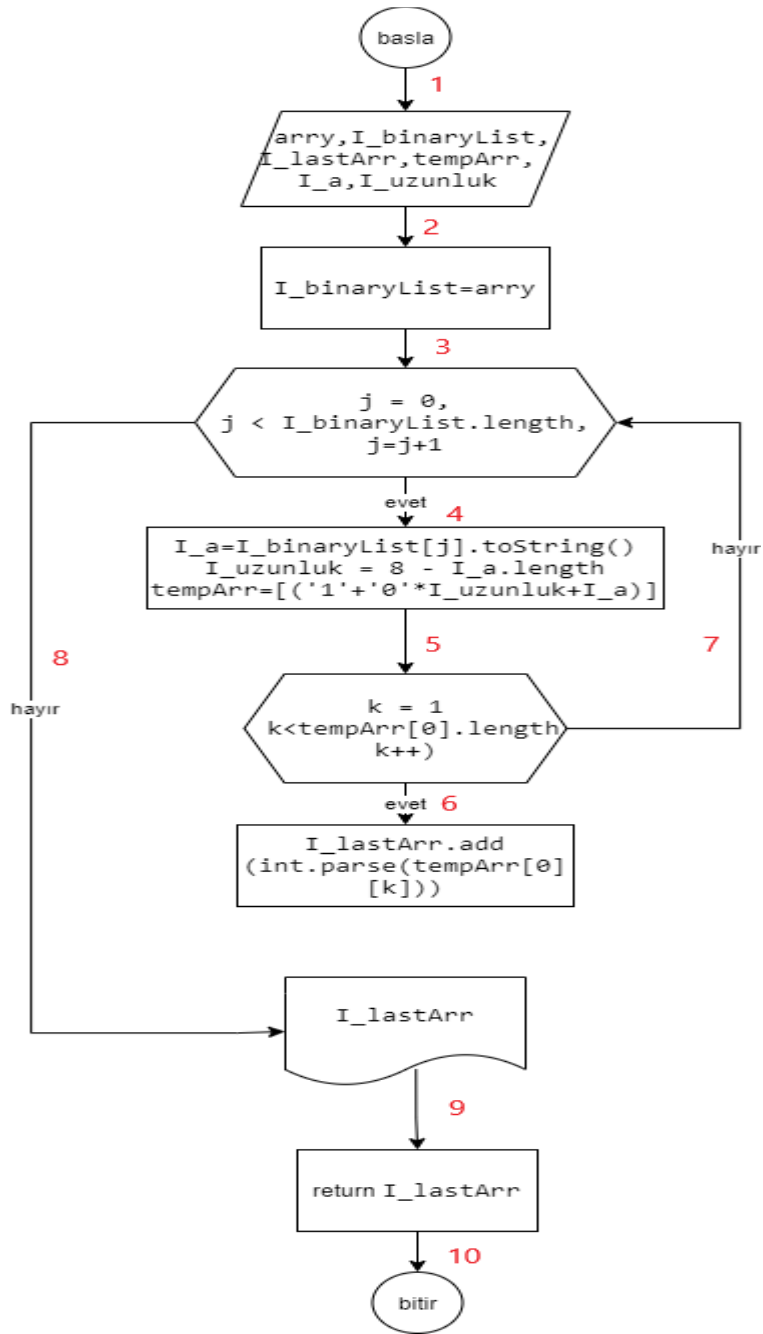
Şekil 3: DecimAltıBinalyFunction Fonksiyonu Akış Şeması

1-2-3

1-2-4-5

Sırasıyla decimalBinalyFunction denendi ve başarıyla çalıştığı görüldü.

4.1.3 DivisionHexFunction();



Şekil 4: DivisionHexFunction Fonksiyonu Akış Şeması

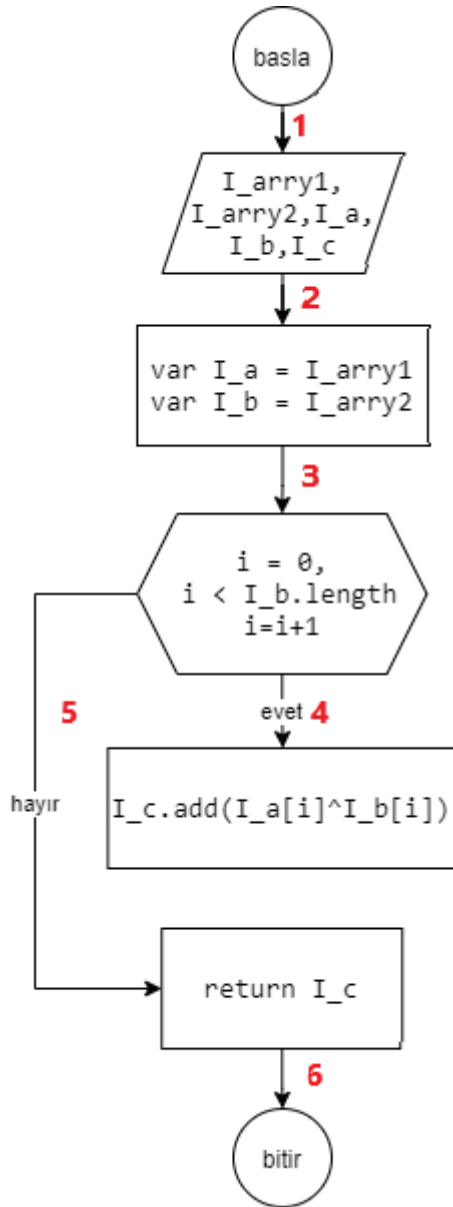
1-2-3-4-5-6

1-2-3-4-5-7

1-2-3-8-9-10

DivisionHexFunction yukarıdaki sıralamayla denedi ve adımları başarıyla çalıştı.

4.1.4 xorFunction();



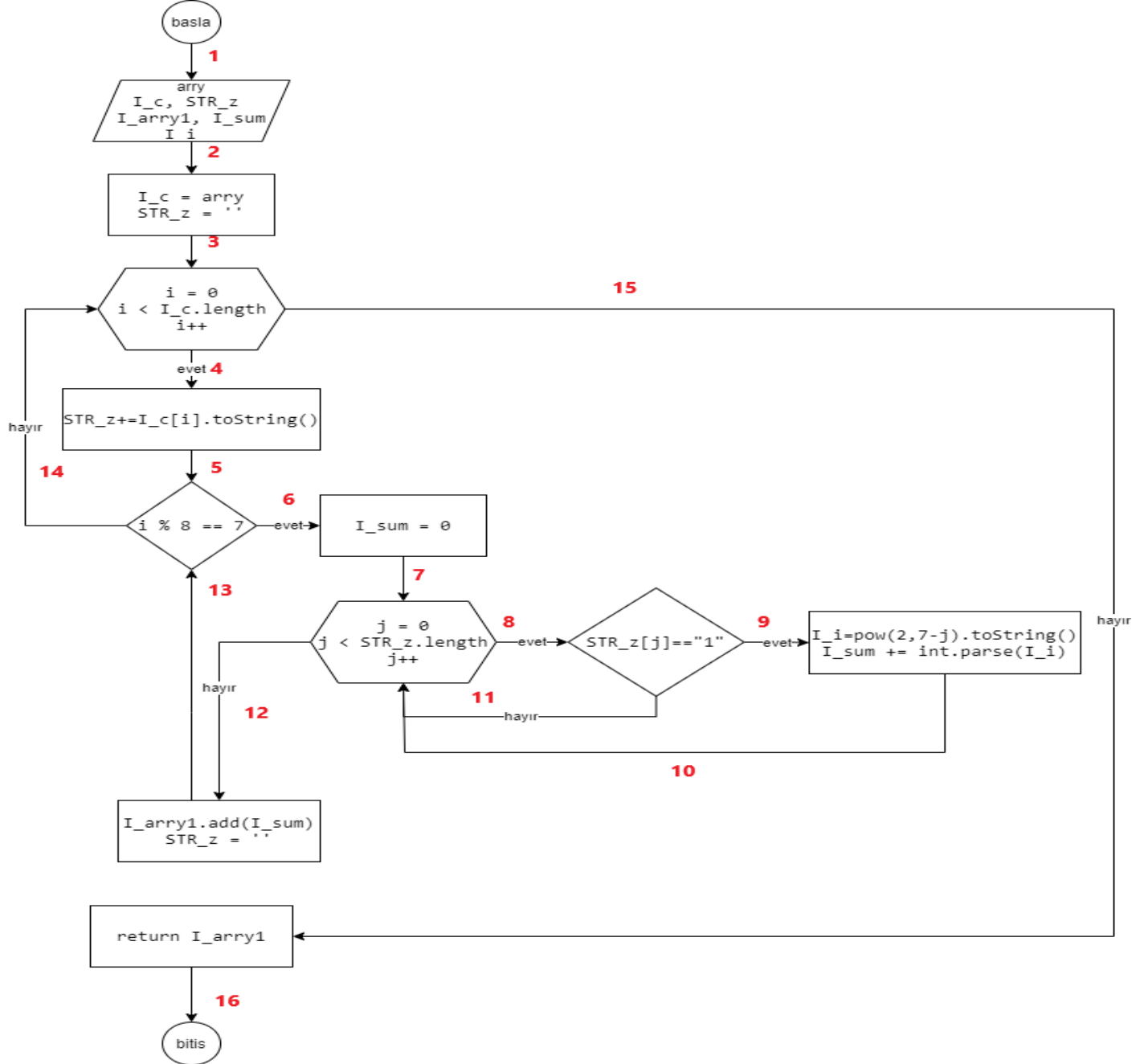
Şekil 5: xorFunction Fonksiyonu Akış Şeması

1-2-3-4

1-2-3-4-6

Adımları denendi ve başarıyla çalıştı.

4.1.5 byteConversionFunction();



Şekil 6: byteConversionFunction Fonksiyonu Akış Şeması

1-2-3-4-5-6-7-8-9-10

1-2-3-4-5-6-7-8-11

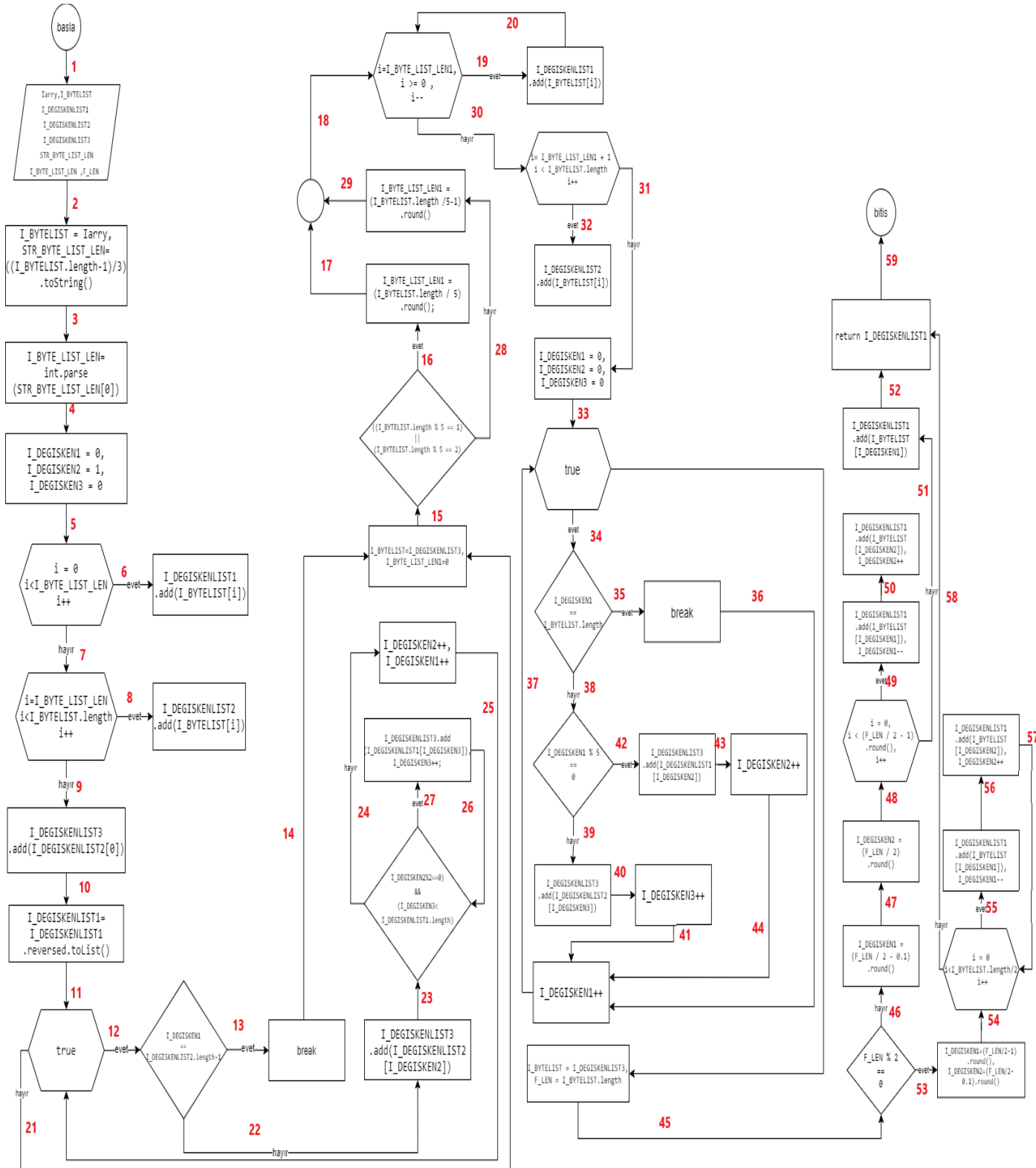
1-2-3-4-5-6-7-12-13

1-2-3-4-5-14

1-2-3-15-16

byteConversionFunction fonksiyonu yukarıda ki adımlara göre çalıştırıldı ve adımlar başarıyla çalıştı.

4.1.6 ReversezigzagFunction();

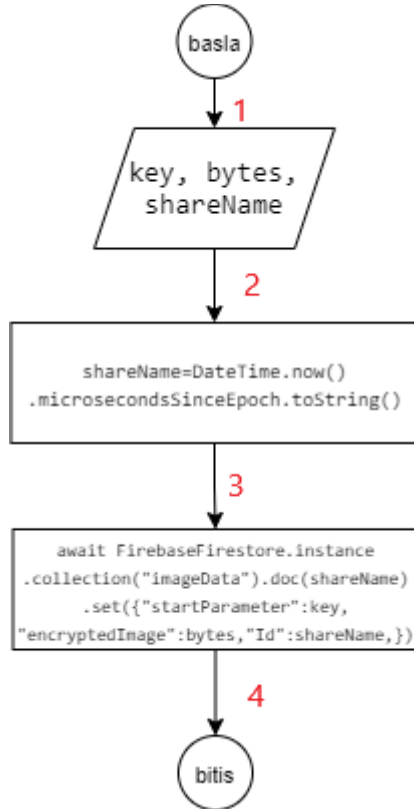


Şekil 7: ReversezigzagFunction Fonksiyonu Akış Şeması

1-2-3-4-5-6
 1-2-3-4-5-6-7-8
 1-2-3-4-5-7-9-10-11-21-15-16-17-18-19-20
 1-2-3-4-5-7-9-10-11-12-13-14-15-16-17-18-19-20
 1-2-3-4-5-7-9-10-11-12-22-23-24-25
 1-2-3-4-5-7-9-10-11-12-22-23-27-26-24-25
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-34-35-36-37
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-34-38-42-43-44-37
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-34-38-40-41-37
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-x-45-46-47-48-49-50
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-x-45-46-47-49-51-52-59
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-x-45-53-54-55-56-57-58-59
 1-2-3-4-5-7-9-10-11-12-13-14-15-28-29-18-30-31-33-x-45-53-54-58-59

ReversezigzagFunction fonkiyonu için yukarıda kibütün adımlar sırasıyla denendi ve bir yanlışlık görülmedi, kodlar başarıyla çalıştı.

4.1.7 ImageToFirestore();

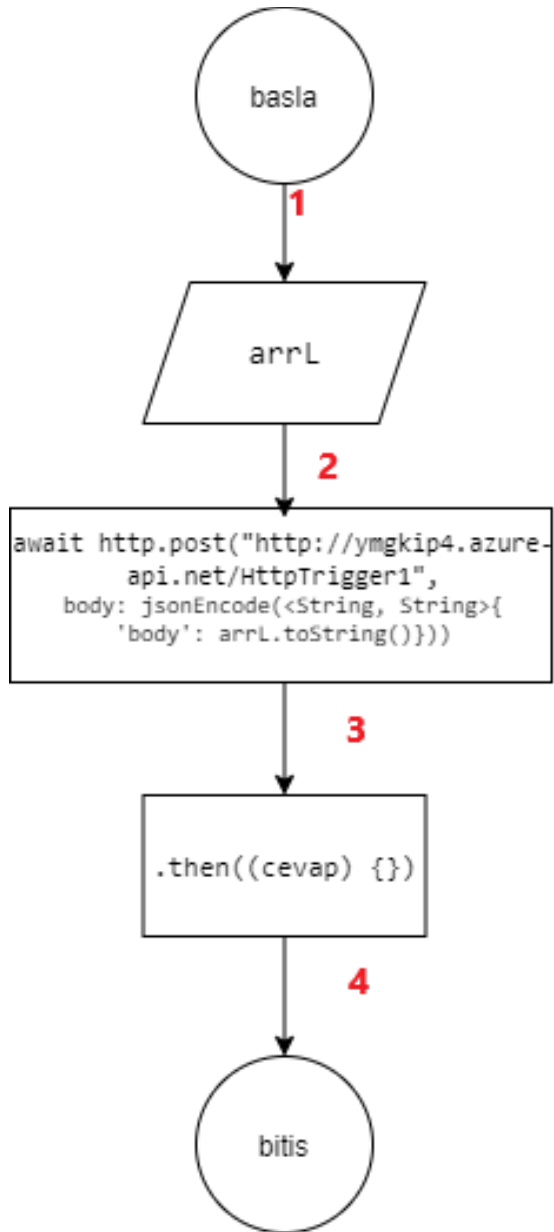


Şekil 8: ImageToFirestore Fonksiyonu Akış Şeması

1-2-3-4

ImageToFirestore() fonksiyonu çalıştırıldı ve başarıyla çalıştı.

4.1.8 PortIslemiYap();

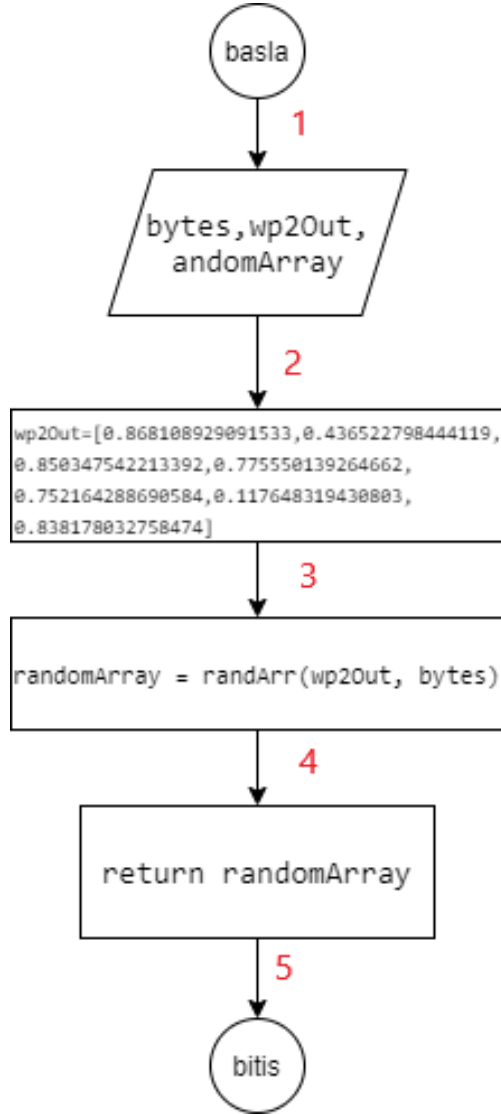


Şekil 9: PortIslemiYap Fonksiyonu Akış Şeması

1-2-3-4

PortIslemislemiYap() fonksiyonu başarıyla çalıştı.

4.1.9 List createSenderKey();

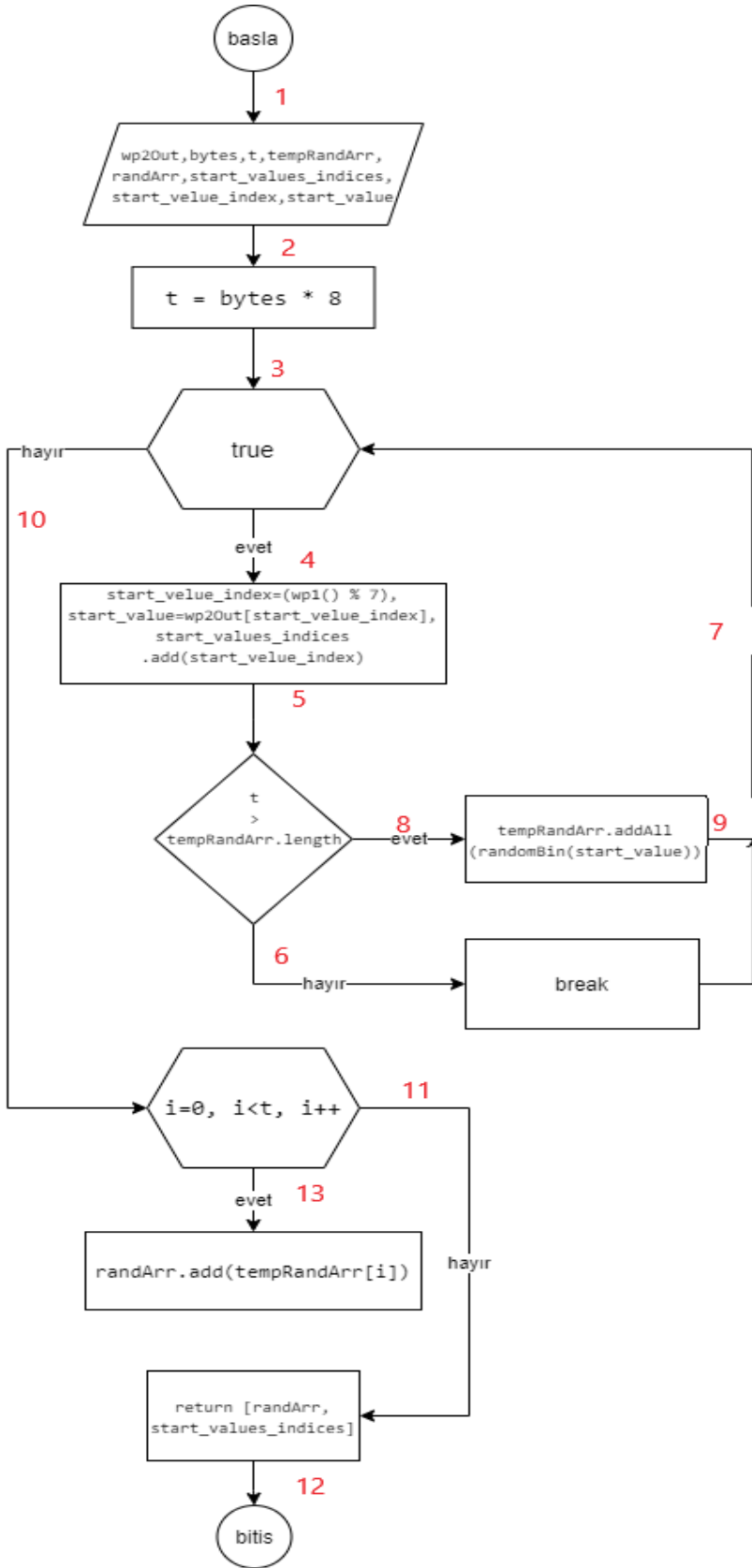


Şekil 10: List createSenderKey Fonksiyonu Akış Şeması

1-2-3-4-5

Sıralamasıyla fonksiyon çalıştırıldı ve bir sorunla karşılaşılmadı.

4.1.10 List randArr();



Şekil 11: List randArr Fonksiyonu Akış Şeması

1-2-3-4-5-8-7

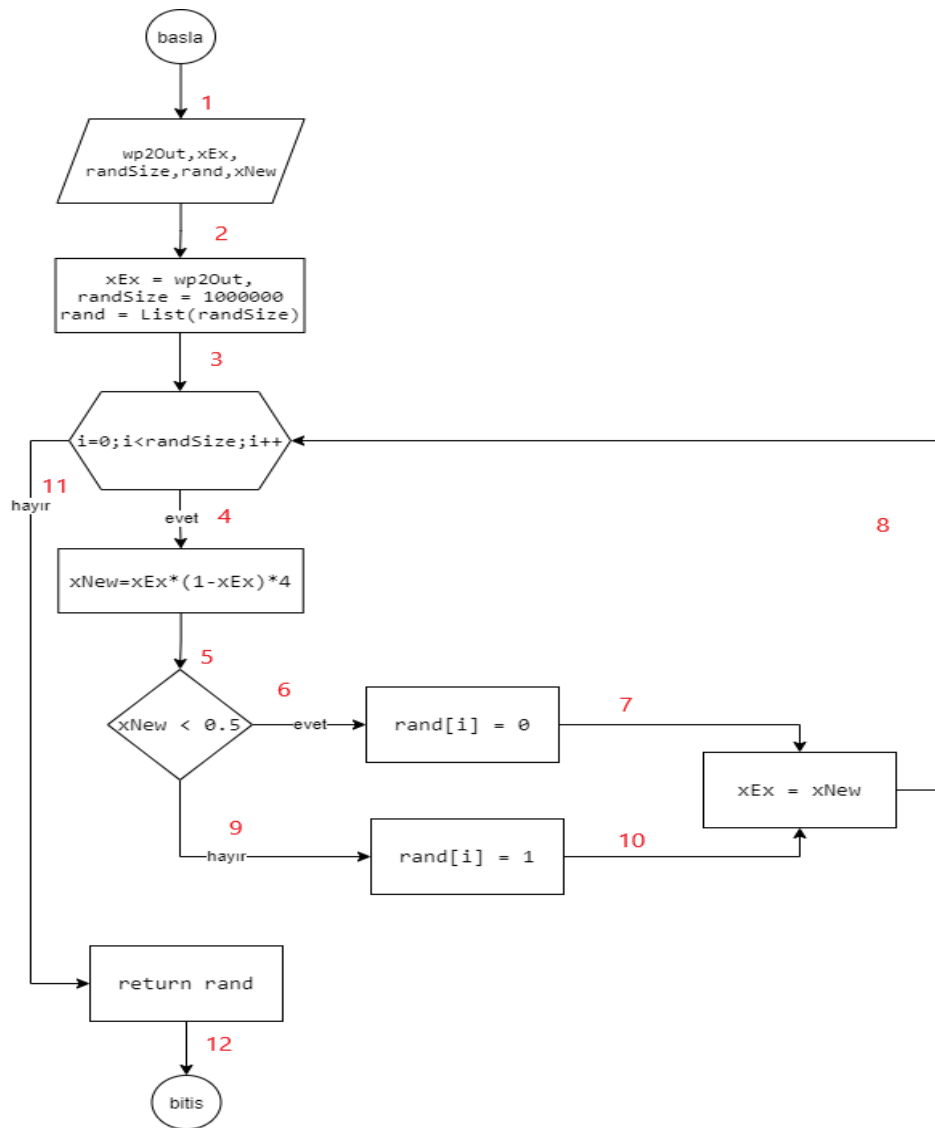
1-2-3-4-5-6-7

1-2-3-10-13

1-2-3-10-11-12

List randArr() fonksiyonu yukarıdaki sıralamalarla test edildi ve başarıyla çalıştı.

4.1.11 List randomBin();



Şekil 12: List randomBin Fonksiyonu Akış Şeması

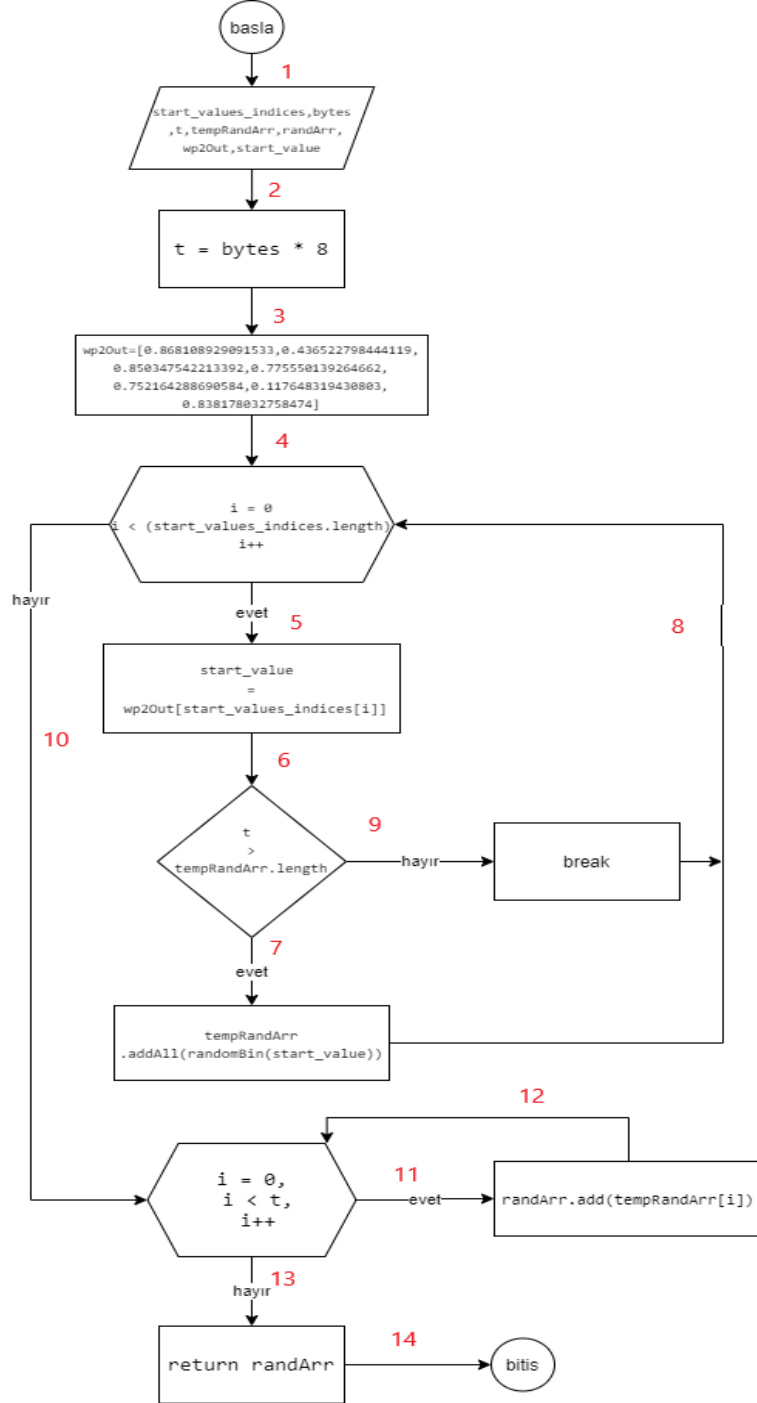
1-2-3-4-5-6-7-8

1-2-3-4-5-9-10-8

1-2-3-11-12

List randomBin fonksiyonu sıralamalarıyla test edildi ve bir sorunla karşılaşılmadı.

4.1.12 List createReceiverKey();



Şekil 13: List createReceiverKey Fonksiyonu Akış Şeması

1-2-3-4-5-6-7-8

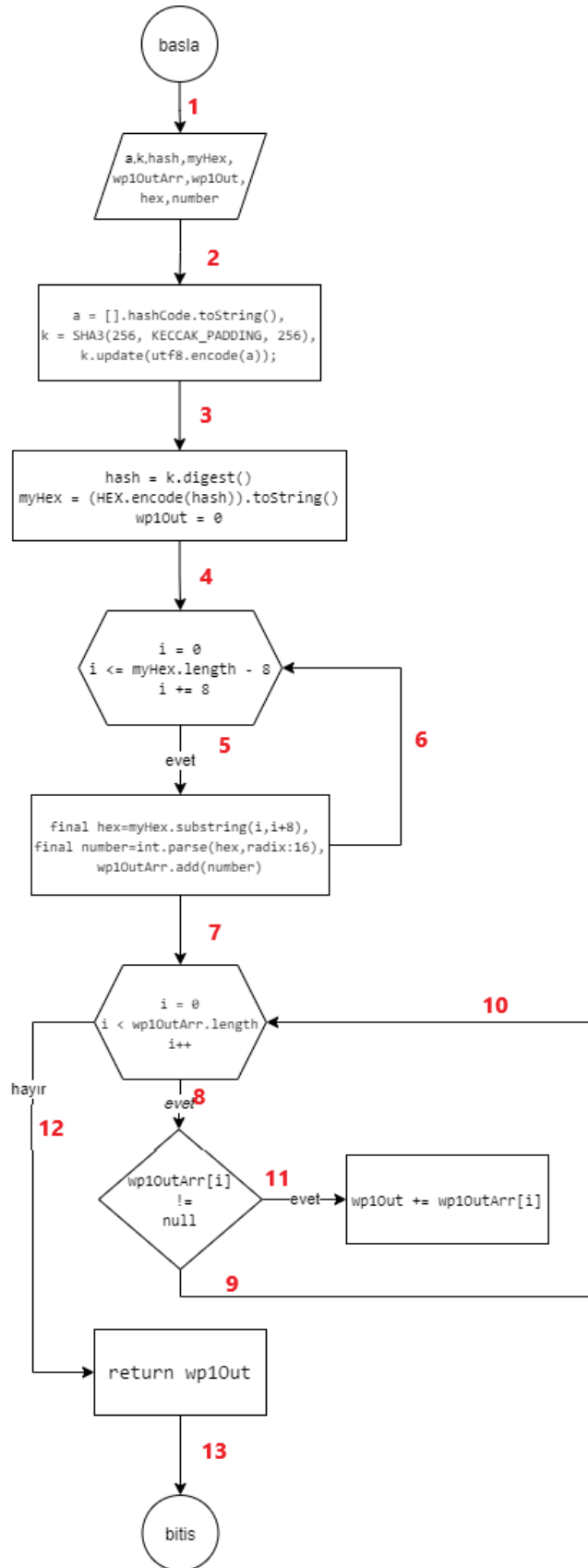
1-2-3-4-5-6-9-8

1-2-3-4-10-11-12

1-2-3-4-10-13-14

List createReceiverKey fonksiyonu için yukarıdaki sıralamayla adımlar gerçekleştirildi, testler başarıyla sağlandı.

4.1.13 int wp10;



Şekil 14: int wp1 Fonksiyonu Akış Şeması

1-2-3-4-5-6

1-2-3-4-5-7-8-11

1-2-3-4-5-8-9

1-2-3-4-5-7-8-9

1-2-3-4-5-7-12-13

int wp1 fonksiyonu yukarıdaki sıralamalarla çalıştırıldı ve her aşamasında başarı sağlandı.

Yukarıda da görüldüğü gibi projenin her fonksiyonu kendi içerisinde ki aşamalarla test edildi ve kodların hepsi başarılı bir şekilde çalıştı. Beyaz kutu testleri başarıyla uygulandı.

5.KALİTE METRİKLERİ

5.1 LCOM

N adet kümenin kesişiminden oluşan kümelerdeki uyumsuzlukların sayısıdır. Metotlardaki benzerlik derecesini ölçer.

Projemizde bulunan fonksiyonlar ve değişkenler tespit edildi;

`zigzagFunction()` → I1 = {array, l_array2, l_array3, l_array4}

`decimaltoBinaryFunction()` → I2 = {array, l_binaryList, l_lastArr}

`divisionHexFunction()` → I3 = {array, l_binaryList, l_lastArr, tempArr, l_a, l_uzunluk}

`xorFunction()` → I4 = { l_array1, l_array2, l_a, l_b, l_c }

`byteconversionFunction()` → I5 = { array, l_c, STR_z, l_array1, l_sum, l_i}

`reversezigzagFunction()` → I6 = {larry, l_BYTELİST, l_DEGİSKENLİST1, l_DEGİSKENLİST2, l_DEGİSKENLİST3, STR_BYTE_LIST_LEN, l_BYTE_LIST_LEN, F_LEN}

`imageToFireStore()` → I7 = { key, bytes, shareName }

`portIslemiYap()` → I8 = { array1}

`List createReceiverkey()` → I9 = { start_valves_indices, bytes, t, temRandArr, randArr, wp20out, start_value}

`List randArr()` → I10 = { wp20out, bytes , t, tempRandArr, randArr, start_valves_indices, start_valve_index, start_valve}

`List randomBin()` → I11 = { wp20out, xEx, randSize, rand, xNew}

`List createSenderKey()` → I12 = { bytes, wp20ut, andomArray}

`int wp1()` → I13 = { a, k, hash, myHex, wp10utArr, wp10ut, hex, number}

Her fonksiyon birbiriyle karşılaştırıldı ve aynı değişkeni kullanan fonksiyonlar tespit edildi;

$(I12 \sim I13) = \emptyset$

$(I11 \sim I13) = \emptyset$

$(I11 \sim I12) = \{wp20ut\}$

$(I10 \sim I13) = \emptyset$

$(I10 \sim I12) = \{start_values_indices, bytes, wp20out, start_value, tempRandArr, t\}$

$(I10 \sim I11) = \{ wp20ut\}$

$(I9 \sim I13) = \emptyset$

(I9 ~ I12)={ wp20ut,bytes }

(I9 ~ I11)={ wp20ut}

(I9 ~ I10)= { wp20ut,bytes }

(I8 ~ I13)= \emptyset

(I8 ~ I12)= \emptyset

(I8 ~ I11)= \emptyset

(I8 ~ I10)= \emptyset

(I8 ~ I9)= \emptyset

(I7 ~ I13)= \emptyset

(I7 ~ I12)={bytes}

(I7 ~ I11)= \emptyset

(I7 ~ I10)= {bytes}

(I7 ~ I9)= {bytes}

(I7 ~ I8)= \emptyset

(I6 ~ I13)= \emptyset

(I6 ~ I12)= \emptyset

(I6 ~ I11)= \emptyset

(I6 ~ I10)= \emptyset

(I6 ~ I9)= \emptyset

(I6 ~ I8)= \emptyset

(I6 ~ I7)= \emptyset

(I5 ~ I13)= \emptyset

(I5 ~ I12)= \emptyset

(I5 ~ I11)= \emptyset

(I5 ~ I10)= \emptyset

(I5 ~ I9)= \emptyset

(I5 ~ I8)= \emptyset

(I5 ~ I7)= \emptyset

(I5 ~ I6)= \emptyset

(I4 ~ I13)= \emptyset

$(l_4 \sim l_{12}) = \emptyset$
 $(l_4 \sim l_{11}) = \emptyset$
 $(l_4 \sim l_{10}) = \emptyset$
 $(l_4 \sim l_9) = \emptyset$
 $(l_4 \sim l_8) = \emptyset$
 $(l_4 \sim l_7) = \emptyset$
 $(l_4 \sim l_6) = \emptyset$
 $(l_4 \sim l_5) = \{l_c, l_{array1}\}$
 $(l_3 \sim l_{13}) = \emptyset$
 $(l_3 \sim l_{12}) = \emptyset$
 $(l_3 \sim l_{11}) = \emptyset$
 $(l_3 \sim l_{10}) = \emptyset$
 $(l_3 \sim l_9) = \emptyset$
 $(l_3 \sim l_8) = \emptyset$
 $(l_3 \sim l_7) = \emptyset$
 $(l_3 \sim l_6) = \emptyset$
 $(l_3 \sim l_5) = \{array\}$
 $(l_3 \sim l_4) = \{l_a\}$
 $(l_2 \sim l_{13}) = \emptyset$
 $(l_2 \sim l_{12}) = \emptyset$
 $(l_2 \sim l_{11}) = \emptyset$
 $(l_2 \sim l_{10}) = \emptyset$
 $(l_2 \sim l_9) = \emptyset$
 $(l_2 \sim l_8) = \emptyset$
 $(l_2 \sim l_7) = \emptyset$
 $(l_2 \sim l_6) = \emptyset$
 $(l_2 \sim l_5) = \{array\}$
 $(l_2 \sim l_4) = \emptyset$
 $(l_2 \sim l_3) = \{array, l_{binaryList}, l_{lastArr}\}$
 $(l_1 \sim l_{13}) = \emptyset$

$$(I1 \sim I12) = \emptyset$$

$$(I1 \sim I11) = \emptyset$$

$$(I1 \sim I10) = \emptyset$$

$$(I1 \sim I9) = \emptyset$$

$$(I1 \sim I8) = \emptyset$$

$$(I1 \sim I7) = \emptyset$$

$$(I1 \sim I6) = \emptyset$$

$$(I1 \sim I5) = \{array, I_array2, I_array3, I_array4\}$$

$$(I1 \sim I4) = \{array\}$$

$$(I1 \sim I3) = \{I_array2\}$$

$$(I1 \sim I2) = \{array\}$$

P değerimiz boş küme olanların sayısı.

Q ise boş küme olmayan sayısıdır.

$$P=60$$

$$Q=18$$

$$LCOM = |P| - |Q|$$

$$LCOM = 60 - 18$$

$$LCOM = 42 \text{ 'dir.}$$

5.2 DÖNGÜSEL KARMAŞIKLIK

Döngüsel karmaşıklık basit bir ifade ile kaynak kodda bulunan karar sayılarını belirlemek için kullanılmaktadır. Bu sayı ne kadar yüksekse sistem o kadar karmaşık olmaktadır.

DK = Y - N + 2P Formülüyle hesaplanır.

Y = Yol, N = Node ve P ise bağlı olmayan yolları ifade eder.

Projemizin döngüsel karmaşıklığına bakalım;

5.2.1 Zigzagfunction();

$$Y=21, N=16$$

$$DK=21-16+2=7$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 7'dir. **DecimalityBinalyFunction();**

$$Y=5, N=6$$

$$DK=5-6+2=1$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 1'dir.

5.2.2 DivisionHexFunction();

$$Y=10, N=10$$

$$DK=10-10+2=2$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 2'dir.

5.2.3 xorFunction();

$$Y=10, N=10$$

$$DK=6-7+2=1$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 1'dir.

5.2.4 byteConversionFunction();

$$Y=16, N=13$$

$$DK=16-13+2=5$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 5'dir.

5.2.5 ReversezigzagFunction();

$$Y=59, N=51$$

$$DK=59-51+2=10$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 10'dir.

5.2.6 ImageToFirestore();

$$Y=4, N=5$$

$$DK=4-5+2=1$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 1'dir.

5.2.7 PortIslemiYap();

$$Y=4, N=5$$

$$DK=4-5+2=1$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 1'dir.

5.2.8 createSenderKey();

$$Y=5, N=6$$

$$DK=5-6+2=1$$

Bu fonksiyonumuzun döngüsel karmaşıklığı 1'dir.

5.2.9 randArr();

Y=12, N=12

DK=12-12+2=2

Bu fonksiyonumuzun döngüsel karmaşıklığı 2'dir.

5.2.10 randomBin();

Y=12, N=11

DK=12-11+2=3

Bu fonksiyonumuzun döngüsel karmaşıklığı 3'dir.

5.2.11 createReceiverKey();

Y=14, N=13

DK=14-13+2=3

Bu fonksiyonumuzun döngüsel karmaşıklığı 3'dir.

5.2.12 wp1();

Y=13, N=11

DK=13-11+2=4

Bu fonksiyonumuzun döngüsel karmaşıklığı 4'dir.

5.3 SLOC ve LSLOC

SLOC açılımı Sources Lines of Code (Kod Satır Sayısı) dur.

LSLOC açılımı Logical Sources Lines of Code (Mantıksal Kod Satır Sayısı) dur

SLOC hesaplanırken yorum satırları göz ardı edilir.

LSLOC ise print, for, while, if-else, vb. yapılar dikkate alınarak hesaplanır.

Projemizde bulunan fonksiyonların SLOC VE LSLOC değerleri;

5.2.1 Zigzagfunction();

SLOC=27

LSLOC=9

5.2.2 DecimalityBinalyFunction();

SLOC=9

LSLOC=1

5.2.3 DivisionHexFunction();

SLOC=14

LSLOC=3

5.2.4 xorFunction();

SLOC=9

LSLOC=1

5.2.5 byteConversionFunction();

SLOC=20

LSLOC=4

5.2.6 ReversezigzagFunction();

SLOC=87

LSLOC=17

5.2.7 imageToFirestore();

SLOC=9

LSLOC=1

5.2.8 PortIslemiYap();

SLOC=10

LSLOC=1

5.2.9 createSenderKey();

SLOC=13

LSLOC=0

5.2.10 randArr();

SLOC=20

LSLOC=4

5.2.11 randomBin();

SLOC=15

LSLOC=3

5.2.12 createReceiverKey();

SLOC=27

LSLOC=4

5.2.13 wp1();

SLOC=19

LSLOC=3

6.MOBİL TESTİ



Mobil uygulamaların test adımları çok sayıda farklı testlerin birleştirilmesinden oluşur. Bunlar;

- Fonksiyonel Testi
- Kullanılabilirlik Testi
- Kullanıcı Arayüz Testi
- Kullanıcı Arayüz Testi
- Uygunluk Testi
- Performans Testi
- Güvenlik testi
- Kurtarma Testi
- Yerelleştirme testi
- Değişiklik testi
- Beta Testi
- Sertifikasyon Testi

Şekil 15: Proje Giriş Ekranı

6.1 Fonksiyonel Testi:

Bu testin amacı gereksinimlerin istenilen şekilde çalışıp çalışmadığıdır, kısaca beklenen işlemleri gerçekleştiriyor mu ? ona bakılır.

Uygulamada incelenen kurallar;

Uygulamanın türüne bakılır(banka, sosyal ağlar, eğitim, vb)

Hedef Kitlelerine bakılır (şirketler ,kullanıcılar ,eğitimciler)

Dağıtım Kanalları incelenir (Google Play, App Store, vb.)



Uygulamanın türü sosyal ağlar ve hedef kitlesi uygulamayı kullanmak isteyen kullanıcılardır, dağıtım kanalı olarak Google Play ve App Store Kullanılmıştır.

Uygulama şekil1 de görüldüğü gibi sorunsuz çalışmaktadır.

Şekil 16: Arayüz1

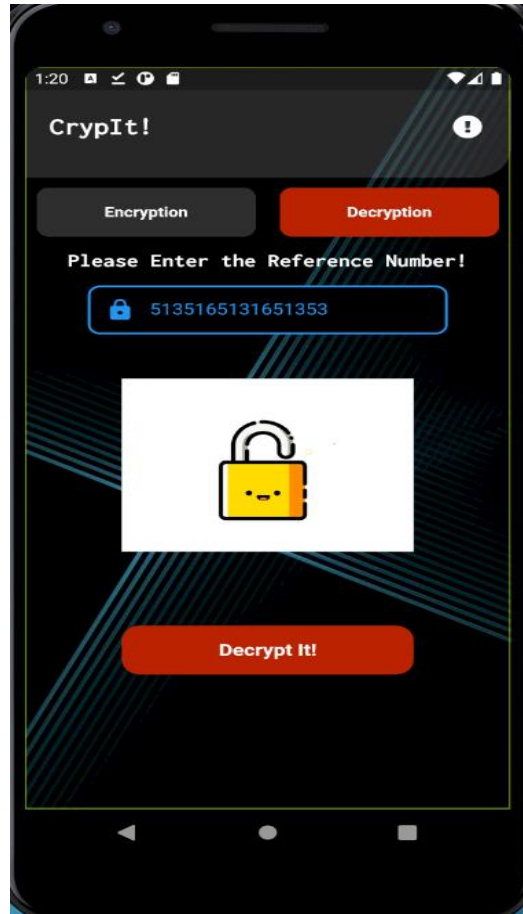
6.2 Kullanılabilirlik Testi:

Uygulamanın kullanılabilir olup olmadığını belirlemek için yapılan test türüdür.

Uygulamada incelenen kurallar

- Butonların normal boyutta olması
- Kullanılan simge ve resimlerin uygulamaya uygun olması
- Uygulamanın hızlı bir şekilde yanıt vermesi

Seçilen resmin orijinal boyut gözetmeksizin uygulamanın içine uygun şekilde sığması sağlandı ve buna uygun butonların boyutları ve konumu ayarlanmıştır. Hız olarak performans testinden anlaşılabileceği üzere hızlı yanıt verebilmektedir.



Şekil17:Arayüz2

6.3 Kullanıcı Arayüz Testi:

Arayüzün istekleri karşılayıp karşılamadığına, uygulamanın arayüzünün standart çözünürlükte çalışıp çalışmadığına bakar

Uygulamada incelenen kurallar

- Farklı cihazlarda performansı incelenir
- Temel tasarım elementleri (butonlar, imgeler, renkler, fontlar, bağlantılar, font boyutları, vb.) test edilir.
- Ekran çözünürlüğüne göre elementlerin durumuna bakar.

Şekil 2 de ve şekil 3 gözüktüğü üzere piksellerin boyutuna göre, butonların yeri ve yazıların fontları arayüze uygun ayarlanmıştır ve kullanımını anlamak için uygulamanın içine kısa kullanım kılavuzu konulmuştur.

Proje mobil uygulama olduğu için test edilen mobil cihazlarda istenilen sonucu kısa zamanda gerçekleştirmiştir.

Proje görüntüyü şifreleme ve karşı tarafa iletmek üzerine olduğu için, karşı tarafa iletmek için seçilen resmin boyutu fark etmeksizin istenilen resim seçilebiliyor ve arayüz olarak kullanıcıyı rahatsız etmeyecek bir tasarım kullanılmıştır.

6.4 Uygunluk Testi:

Testin amacı, uygulamanın farklı cihazlarda da en iyi performansı göstermesini sağlamaktır.

Uygulamada incelenen kurallar

- İşletim sistemi
- Tarayıcıya
- Veritabanına
- Cihaza
- Bağlantı ayarlarına

Proje android ve ios işletim sisteminde istenilen performansı göstermiştir, yani iki işletim sisteminde sağlıklı bir şekilde çalışmaktadır.

6.5 Performans Testi:

Performans testi uygulamanın stabil olup olmadığına bakar, farklı farklı senaryolarda ve yükler ile sistem test edilir.

Uygulamada incelenen kurallar

- Uygulamanın stabil olarak çalıştırılır ve performansı incelenir
- Farklı stresler uygulayarak (farklı boyutta resimler, uygulamanın yanında arka planda yeni uygulamalar açarak vb.) performansı ölçülür

Daha önceden kodlara performans testi uygulayarak kodların performans testlerden başarılı bir şekilde geçmiştir.

Projenin amacına göre seçilen resmin boyut veya çözünürlük fark etmeksizin bütün mobil cihazlarda ve mobil işletim sistemlerinde görüntüyü kısa sürede istenildiği gibi resmi şifrelemekte ve karşı tarafa iletmektedir.

Bu testi uygularken farklı boyutlarda ve farklı telefonlarda denenmiş ve başarıyla geçmiştir

6.6 Güvenlik testi:

Güvenlik Testleri, yazılım sistemlerinin kötü niyetli kullanımı sebebiyle oluşabilecek sorunların önceden tespiti amacıyla gerçekleştirilen test çeşididir.

Uygulamada incelenen kurallar

- Giriş, şifre, banka kart numaraları, vb. hassas verilerin ağ saldırılarına karşı güvende olup olmadığına bakılır.

Uygulama karşı tarafa şifrelenmiş resim gönderme üzerine olduğundan ötürü yazılan kodların algoritması tamamıyla her seferinde farklı değerler üretecek şekilde yazılmıştır bundan ötürü hacklenmesi zorlaşmıştır.

Bunun yanında uygulama özel bir kullanıcı adı, şifre veya hassas veri (banka kart numarası) gibi durumlar gerektirmemektedir. Bunun akabinde bu tür saldırılara sistemimiz kapalıdır.

6.7 Kurtarma Testi:

Bu testin amacı herhangi bir hata durumunda verilerin kurtarılmasını sağlamaktır.

Uygulamada incelenen kurallar

- Bağlantıda gerçekleşen bir kopmanın ardından veri sağlam bir şekilde duruyor mu? Bakılır.
- Uygulama aniden kapanması durumunda verinin son durumuna bakılır.
- Telefonun aniden kapanması sonucunda verinin durumuna bakılır.

////////////////////////////////////
////////////////////////////////////

GRUP BAŞKANI

6.8 Yerelleştirme testi:

Uygulamanın belirli bir coğrafyada belirli kültüre hitap edip etmediğine bakılır.

Uygulamada incelenen kurallar

- Uygulamanın dil seçeneğine bakılır.
- Çevirilerin doğru olup olmadığına bakılır.
- Tarih formatına bakılır.

Uygulamada dünyada kabul edilmiş ortak dil olan İngilizce kullanılmıştır, arayüzlerde kullanılacak kısımlar ve açıklamalar İngilizce olarak belirtilmiştir.

6.9 Değişiklik testi:

Test aşamasında ortaya çıkan bugları temizlemek için çoğu zaman uygulamanın değişmesi ya da güncellenmesi gerekmektedir. Bu testin amacı da yapılan bu değişikliklerin sistemin çalışan parçalarını etkilememesini sağlamaktır.

Uygulamada incelenen kurallar

- Ortaya çıkan tüm bugların düzeltilip düzeltilmediği kontrol edilir
- Ayrıca düzeltilen bugların yeni hataların çıkmasına yol açmayacağından emin olunur.

Tasarım aşamasında ve proje gerçekleştikten sonra bütün buglar düzeltilmiş ve hatasız olarak çalışmaktadır.

6.10 Beta Testi:

Çalışan bir uygulamada yapılır amaç son sürümden önce hataları belirlemek ve düzeltmektir.

Bundan önce bütün testleri sağlıklı bir şekilde geçen uygulama şuan kullanılabilir haldedir, hiçbir hata bulunmamaktadır.

6.11 Sertifikasyon Testi:

.apk uzantılı dosyalar için belirli kurallar bulunmaktadır ve bu kurallara uyulması gerekmektedir. Bu uygulamalar ardından mağazalara yüklenir ve sertifikasyon problemi olmayan uygulamalar mağazalarda yerini alır.

////////////////////////////////////
////////////////////////////////////

GRUP BAŞKANI