BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini akan membahas penggunaan sistem informasi penghasil berkas laporan yang meliputi implementasi dan pengujian perangkat lunak yang dibuat. Implementasi ini digunakan untuk menjelaskan dan mendeskripsikan bagian-bagian yang ada pada sistem. Sedangkan pengujian digunakan untuk menganalisis apakah sistem yang dibuat sudah memenuhi target yang ingin dicapai.

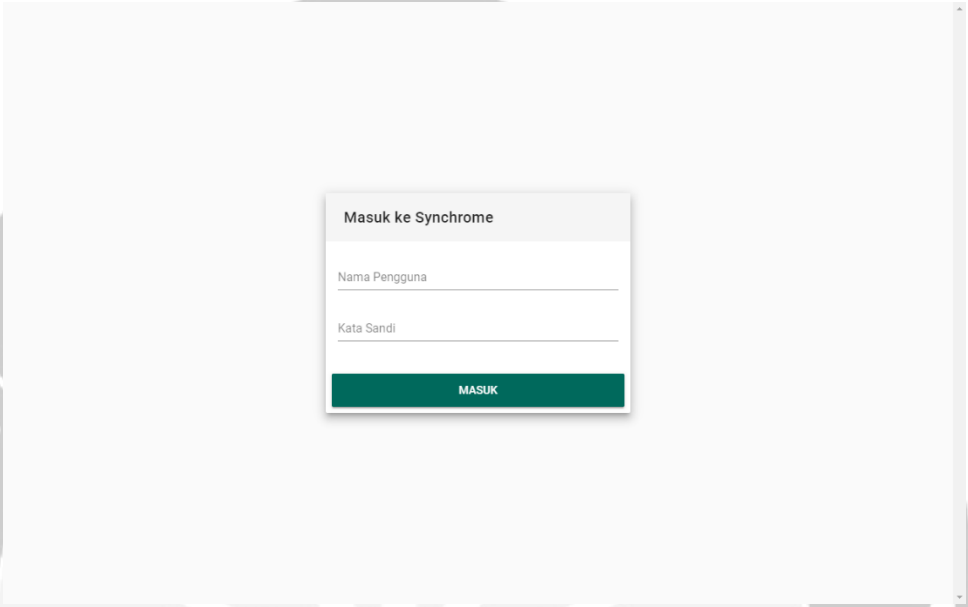
5.1 Definisi Perangkat Lunak

Synchrome merupakan sistem yang terdiri dari 3 komponen terpadu yang memiliki fungsi utama untuk menghasilkan laporan kehadiran ASN beserta besaran TPP yang diterima oleh ASN tersebut dalam bentuk berkas berekstensi .pdf. Sistem ini terdiri dari 2 *platform*, yakni *Synchrome API* dan *Synchrome Web* yang berjalan dalam *platform* web dan *Synchrome Desktop* yang berjalan dalam *platform desktop*. Data utama dikelola oleh *Admin*, sedangkan peran *User* hanya dapat mencetak laporan saja. Dalam pengembangannya, *Synchrome API* dibangun dengan bahasa PHP 7.2 dengan kerangka Lumen 5.7. *Synchrome Web* dan *Synchrome Desktop* dibangun dengan bahasa JavaScript dengan *syntax* ES2015 dengan kerangka Vue. *Synchrome Web* ditranspilasi untuk dapat berjalan pada hampir seluruh browser dan *Synchrome Desktop* ditranspilasi dan dikompilasi menjadi bentuk *executeable* dan menjadi aplikasi *desktop* yang dapat berjalan tanpa konektivitas internet.

5.2 Implementasi Sistem

5.2.1 Synchrome Web

1. Otentikasi



Gambar 5.1 – Antarmuka Halaman Otentikasi

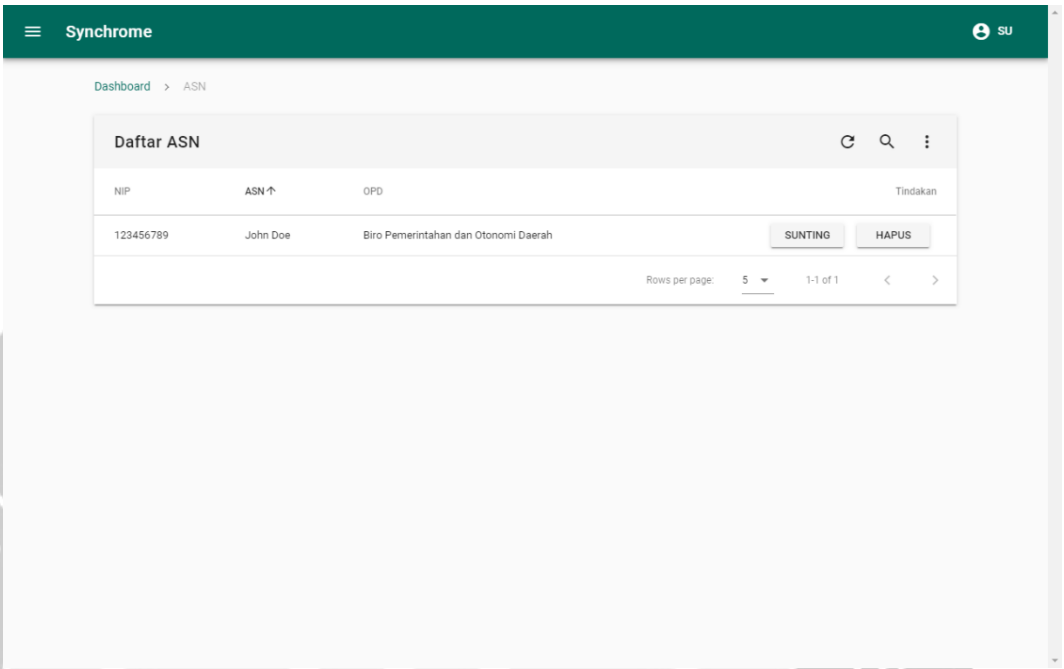
Gambar 5.1 menunjukkan antarmuka halaman otentikasi. Pengguna perlu menginputkan nama pengguna dan kata sandinya untuk kemudian dicek oleh sistem. Apabila proses otentikasi berhasil, pengguna akan di-*redirect* ke halaman *Dashboard*, jika gagal maka pengguna akan diberikan peringatan bahwa nama pengguna/kata sandi salah. Berikut adalah potongan kode untuk proses otentikasi:

```
public function authenticate($name, $password): AccessToken
  {
    try {
      $user = User::where('name', $name)->firstOrFail();

      if (! password_verify($password, $user->password)) {
        throw new InvalidCredentialsException;
      }

      $now = Carbon::now();
```

```
            $access_token_data = [
                'username' => Str::random(32),
                'password' => Str::random(32),
                'generated_at' => $now,
                'last_used_at' => $now,
            ];

            $access_token = $user->accessTokens()->create($access_token_data);

            return $access_token;
        } catch (ModelNotFoundException $e) {
            throw new InvalidCredentialsException;
        } catch (\Exception $e) {
            throw $e;
        }
    }
```

Otentikasi pada *Synchrome API* dilakukan secara *stateless* yang berarti *server* tidak akan menyimpan data apa pun tentang *client* yang telah terotentikasi saat ini (*Session*). Proses dimulai dengan mengecek apakah ada pengguna yang sesuai dengan nama pengguna yang diinputkan. Jika tersedia, maka kata sandi akan dibandingkan. Apabila otentikasi berhasil, sistem akan membuat *access token* yang dapat digunakan untuk mengakses *Synchrome API* dengan metode *HTTP Basic*.

2. Manajemen ASN

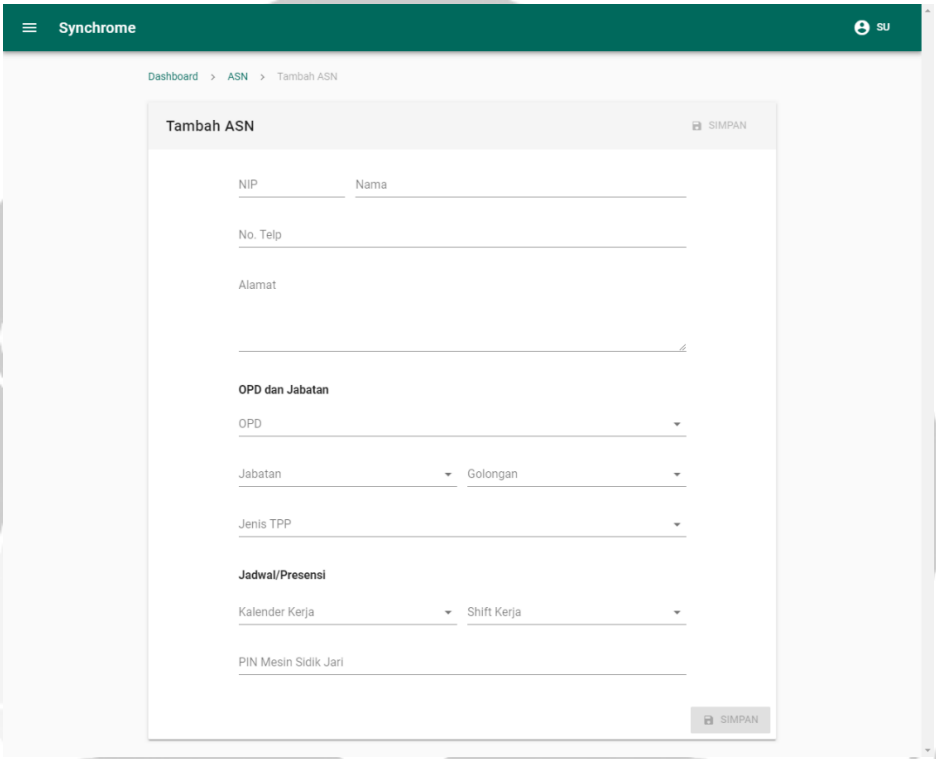


Gambar 5.2 – Antarmuka Halaman Daftar ASN

Gambar 5.2 menunjukkan antarmuka halaman daftar ASN. Pada halaman ini data ASN akan ditampilkan dalam bentuk tabel. Pengguna juga dapat melakukan *refresh* data, pencarian, dan navigasi ke halaman tambah/sunting ASN, menghapus ASN. Berikut potongan kode untuk menampilkan ada ASN:

```
protected $includes = [
    'agency',
    'rank',
    'echelon',
    'tpp',
    'workshift',
    'calendar',
    'fingerprints',
];

public function get()
{
```

```
    return Asn::with($this->includes)->get();
}
```



Gambar 5.3 – Antarmuka Halaman Penambahan/Penyuntingan Data ASN

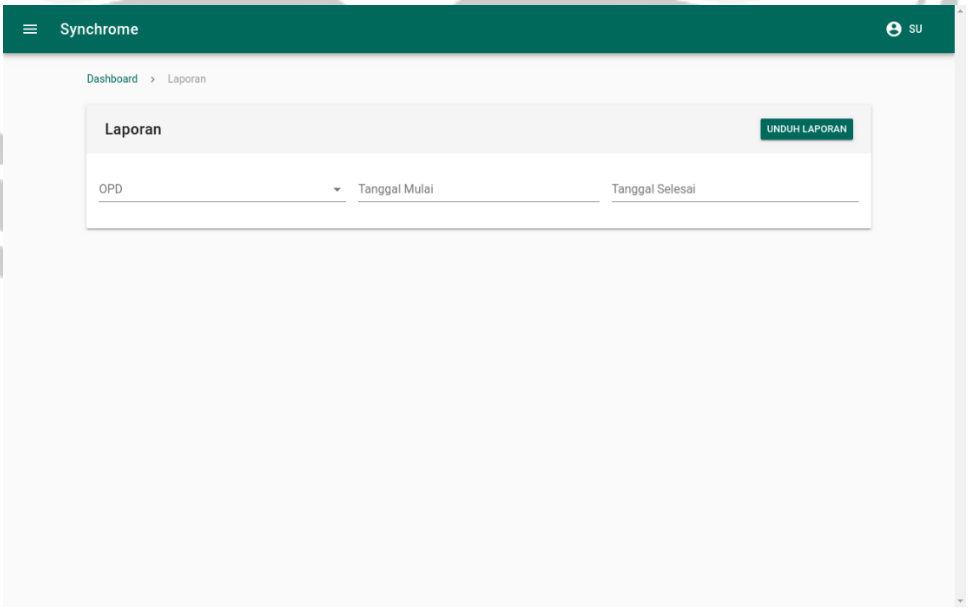
Gambar 5.3 menunjukkan antarmuka halaman untuk penambahan/penyuntingan data ASN. 2 halaman ini menggunakan formulir yang sama, hanya diakomodir prosesnya oleh *route* yang berbeda. Proses penambahan/pembaruan akan dilakukan ketika seluruh data sudah diisi dengan valid, tombol Simpan akan aktif (dapat ditekan). Ketika tombol Simpan ditekan, sistem akan melakukan proses penambahan/pembaruan. Berikut adalah potongan kode untuk salah satu proses dalam halaman ini, yakni penyimpanan data ASN:

```
public function create(array $data)
{
        return Asn::create([
```

```
            'id' => $data['id'],
            'agency_id' => $data['agency_id'],
            'rank_id' => $data['rank_id'],
            'echelon_id' => $data['echelon_id'],
            'tpp_id' => $data['tpp_id'],
            'workshift_id' => $data['workshift_id'],
            'calendar_id' => $data['calendar_id'],
            'pin' => $data['pin'],
            'name' => $data['name'],
            'phone' => $data['phone'],
            'address' => $data['address'],
        ]);
    }
```

Setelah inputan divalidasi pada level *controller*, *service layer* akan menyimpan data ASN dengan menggunakan *model* yang telah dibuat sebelumnya.

3. Pengadaan Laporan



Gambar 5.4 – Antarmuka Halaman Pengadaan Laporan

Gambar 5.4 menunjukkan antarmuka halaman pengadaan laporan. Pada halaman ini pengguna dapat menginputkan OPD, tanggal mulai, dan

tanggal selesai untuk laporan yang diinginkan. Sistem kemudian akan mengambil data setiap ASN untuk kemudian dikalkulasikan TPP-nya. Berikut potongan kode untuk proses kalkulasi TPP pada sebuah OPD:

```
public function getAttendancesOn(array $data)
{
   try {
      $start = Carbon::parse($data['start']);
      $end = Carbon::parse($data['end']);

      $asn_list = Asn::where('agency_id', $data['agency_id'])
         ->with([
            'tpp',
         ])
         ->get();

      $attendance_map = Attendance::whereBetween('date', [$data['start'], $data['end']])
         ->with([
            'type',
         ])
         ->get()
         ->reduce(function ($carry, $item) {
            $carry[$item->asn_id] = isset($carry[$item->asn_id])
               ? array_merge($carry[$item->asn_id], [$item])
               : [$item];

            return $carry;
         }, []);

      return $asn_list->map(function ($asn) use (
         $attendance_map,
         $start,
         $end
```

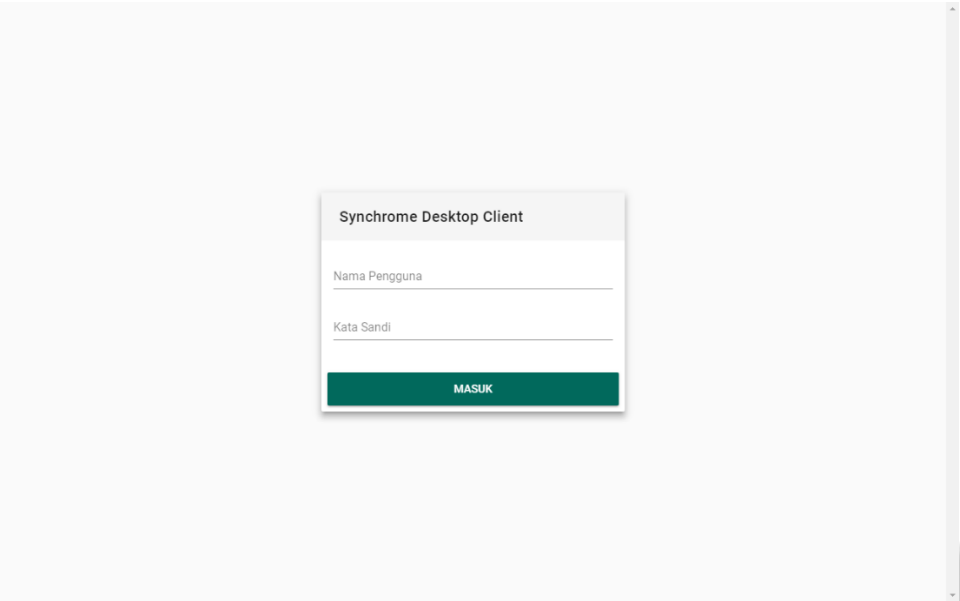
```
    ) {
        $attendances = $attendance_map[$asn->id] ? $attendance_map[$asn->id] : null;
        $initial_tpp = $asn->tpp->value;

        if (! $attendances) {
            return [
                'initial_tpp' => $initial_tpp,
                'final_tpp' => 0,
                'attendance_rate' => 0,
            ];
        }

        $expected_total_work_duration = $asn->getTotalWorkDurationBetween(
            $start->format('Y-m-d'),
            $end->format('Y-m-d')
        );

        $actual_total_work_duration = collect($attendances)
            ->filter(function ($item) {
                return $item->type->tpp_paid;
            })
            ->reduce(function ($carry, $item) {
                $today = date('Y-m-d');
                $check_in = Carbon::parse($today . ' ' . $item->check_in);
                $check_out = Carbon::parse($today . ' ' . $item->check_out);

                return $carry + ($check_out->diffInMinutes($check_in));
            }, 0);

        $attendance_rate = $actual_total_work_duration /
$expected_total_work_duration;
        $final_tpp = $attendance_rate * $initial_tpp;
```

```
            return [
                'initial_tpp' => $initial_tpp,
                'final_tpp' => $final_tpp,
                'attendance_rate' => $attendance_rate * 100,
            ];
        });
    } catch (\Exception $e) {
        throw $e;
    }
}
```

Proses pembuatan laporan TPP dimulai dari menyiapkan data awal seperti daftar ASN yang berada pada OPD yang hendak dibuatkan laporan, menyiapkan *associative array* data kehadiran dengan NIP ASN sebagai kunci untuk akses yang lebih cepat ketika proses kalkulasi. Kemudian, sistem akan menghitung durasi jam kerja normal dari ASN tersebut berdasarkan *shift* kerja dan kalender kerjanya, kemudian menghitung durasi jam tidak kerja (terlambat/absensi). Perhitungan ini dilakukan dalam satuan menit. Persentase dari selisih dari durasi kerja normal dan durasi tidak bekerja akan dikalikan dengan besaran TPP yang harusnya didapat oleh ASN tersebut. Laporan kemudian akan dibuat berdasarkan data perhitungan tadi dan sistem akan mengembalikan berkas dalam ekstensi .pdf.

5.2.2 Synchrome Desktop

1. Otentikasi



Gambar 5.5 – Antarmuka Halaman Otentikasi *Synchrome Desktop*

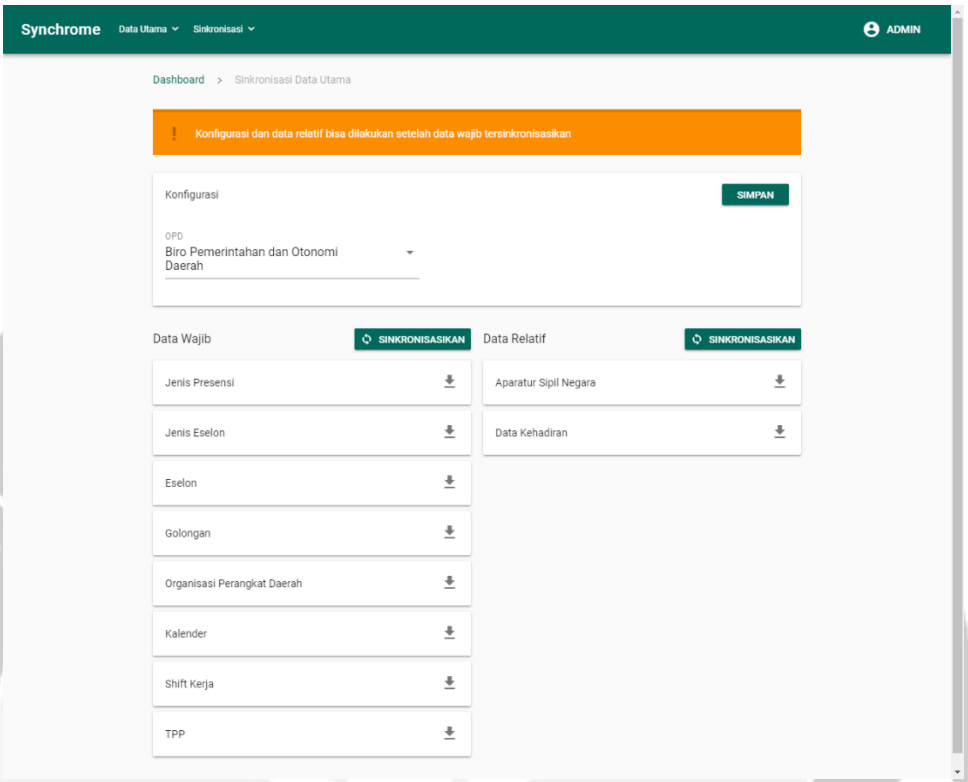
Gambar 5.5 menunjukkan halaman antarmuka otentikasi. Otentikasi pada *Synchrome Desktop* memiliki sumber data yang berbeda. *Synchrome Desktop* memiliki basis data tersendiri, sehingga pengecekan otentikasi di sini dilakukan pada basis data lokal. Berikut potongan kode untuk proses otentikasi pada *Synchrome Desktop*:

```
async authenticate (name, password) {
  try {
   const users = await this._db.User.findAll({
    where: {
     name
    },
    limit: 1
   })

   if (users.length === 0) {
    throw new Error('Pengguna tidak ditemukan')
   }
```

```
const passwordCorrect = this._bcrypt.compareSync(password, users[0].password)

if (!passwordCorrect) {
  throw new Error('Kata sandi salah')
}

return {
  id: users[0].id,
  name: users[0].name,
  createdAt: users[0].createdAt,
  updatedAt: users[0].updatedAt
}
} catch (err) {
  throw err
 }
}
```

Proses otentikasi ini tidak jauh berbeda dengan proses pada *Synchrome API*. Hanya saja kode ini diimplementasi dengan bahasa *JavaScript* dan balikan dari fungsi ini bukan *access token*, melainkan obyek dari pengguna yang terotentikasi saat ini.

2. Sinkronisasi Data



Gambar 5.6 – Antarmuka Halaman Sinkronisasi Data

Gambar 5.6 menunjukkan antarmuka halaman sinkronisasi data. Sinkronisasi terbagi menjadi 2 bagian, Data Wajib dan Data Relatif. Data Wajib akan berupa data yang cenderung sama dimiliki oleh setiap *Synchrome Desktop* yang terinstal pada OPD-OPD, sedangkan Data Relatif akan berupa data yang bersifat khusus untuk tiap OPD seperti data ASN pada OPD tersebut dan data kehadiran yang diambil dari mesin pemindai sidik jari. Berikut potongan kode untuk *sync module* dari salah satu *item*:

```
export default async function (syncModule, axios, db) {
 const model = db[syncModule.model]
 if (!model) return

 try {
  const agencyIdConfig = await db.Config.findOne({
   where: {
```

```
    key: 'agencyId'
  },
  attributes: ['key', 'value']
})

if (!agencyIdConfig) return

const url = syncModule.target.replace('{id}', agencyIdConfig.value)
const { data } = await axios.get(url)
const formattedData = removeDataNamespace(data.data)

for (let item of formattedData) {
  console.info(`Processing ${syncModule.model} ${item.id}`)

  const foundData = await model.findByPk(item.id)
  const payload = {
    ...item,
    agencyId: item.agency
      ? item.agency.id
      : null,
    rankId: item.rank
      ? item.rank.id
      : null,
    echelonId: item.echelon
      ? item.echelon.id
      : null,
    tppId: item.tpp
      ? item.tpp.id
      : null,
    workshiftId: item.workshift
      ? item.workshift.id
      : null,
    calendarId: item.calendar
```

```
    ? item.calendar.id
    : null
}

if (!foundData) {
  await model.create(payload)
} else {
  if (moment(foundData.updatedAt).isBefore(item.updatedAt)) {
    await model.update(payload, {
      where: {
        id: formattedData.id
      }
    })
  }
}

const fingerprintModel = db.Fingerprint
if (!fingerprintModel) return

for (let fingerprintData of item.fingerprints) {
  const fingerprintFoundData = await fingerprintModel.findByPk(fingerprintData.id)
  const fingerprintPayload = {
    id: fingerprintData.id,
    asnId: item.id,
    idx: fingerprintData.idx,
    algVer: fingerprintData.algVer,
    template: fingerprintData.template
  }

  if (!fingerprintFoundData) {
    await fingerprintModel.create(fingerprintPayload)
  } else {
```

```
        if (moment(fingerprintFoundData.updatedAt).isBefore(fingerprintData.updatedAt))
{
          await fingerprintModel.update(fingerprintPayload, {
            where: {
              id: fingerprintData.id
            }
          })
        }
      }
    }
  } catch (err) {
    throw err
  }
}
```

Potongan kode di atas merupakan *sync module* untuk data ASN. Dimulai dari mencocokkan apakah *model* ASN tersedia atau tidak, jika tidak maka proses akan dilewati. Kemudian akan dilanjutkan dengan mengambil data konfigurasi OPD pada *Synchrome Desktop* untuk mengambil data ASN sesuai dengan OPD-nya. Berikutnya, sistem akan melakukan *request* HTTP ke *Synchrome API* dan melakukan iterasi pada setiap *item* yang diterima. Pada iterasi ini, sistem akan menyiapkan obyek *JavaScript* yang akan menjadi *payload* untuk menyimpan/memperbarui data ASN di basis data *Synchrome Desktop*. Untuk membandingkan data, sistem akan menggunakan *timestamp* yang tersedia pada data tersebut untuk membandingkan dengan data yang saat ini tersimpan pada basis data. Apabila data yang datang tersebut belum ada di basis data, maka sistem akan menyimpannya, jika ternyata sudah ada dan data yang datang lebih baru, maka sistem akan memperbarui basis datanya.

5.3 Hasil Pengujian

Pengujian *unit test* dilakukan pada bagian *back-end* (*Synchrome API*) yang merupakan pusat pengolahan data. Untuk menjalankan pengujian, sistem memerlukan *dependency* tambahan yakni PHPUnit. Tolak ukur keberhasilan pengujian ini adalah ekspektasi keluaran dari setiap *service layer* pada *Synchrome API*. Penulisan *test* akan dibentuk dengan format yang menyesuaikan standar PHPUnit.

Ketika melakukan pengujian, tidak seluruh komponen sistem akan dijalankan secara normal. Beberapa komponen akan dipersiapkan dengan data-data tertentu sehingga pengujian dapat dilakukan dengan lebih spesifik pada suatu fungsi. Konfigurasi data-data awal (*seed*) ini akan dilakukan pada setiap *test* tepat sebelum *test-test* tersebut dijalankan. Berikut adalah contoh potongan kode untuk melakukan konfigurasi data awal pada *test* yang dilakukan pada *AsnService*:

```
public function setUp()
{
    parent::setUp();

    $seeder = app(DatabaseSeeder::class);
    $seeder->call('AgenciesTableSeeder');
    $seeder->call('RanksTableSeeder');
    $seeder->call('EchelonsTableSeeder');
    $seeder->call('TppTableSeeder');
    $seeder->call('WorkshiftsTableSeeder');
    $seeder->call('AttendanceTypesTableSeeder');
    $seeder->call('CalendarsTableSeeder');
    $seeder->call('AsnTableSeeder');

    $this->faker = Factory::create();
    $this->test_asn_service = new AsnService;
}
```

Potongan kode di atas bertujuan untuk memberikan konfigurasi awal pada data OPD, Golongan, Eselon, TPP, *Shift* Kerja, Jenis Kehadiran, Kalender, dan ASN. Kode di atas juga mempersiapkan obyek *AsnService* yang akan digunakan sepanjang *test* dilakukan.

Penulisan *test* akan dilakukan secara spesifik pada suatu fungsi. Pada contoh berikutnya, fungsi yang akan diuji adalah *method* get() pada kelas *AsnService*. Penulisannya akan dilakukan seperti contoh berikut:

```
public function testGet()
{
   $asns = $this->test_asn_service->get();

   $this->assertInstanceOf(Collection::class, $asns);
   $this->assertCount(1, $asns);
}
```

Pengujian dilakukan dengan menerima keluaran dari *method* get() pada kelas *AsnService*. Keluaran akan diuji apakah benar merupakan obyek dari kelas *Collection* (larik) dan diuji apakah benar jumlah data dalam larik tersebut berjumlah satu buah (berdasarkan data yang telah dipersiapkan sebelumnya). Ketika salah satu dari dua *assertion* tersebut tidak terpenuhi maka pengujian akan dianggap gagal.

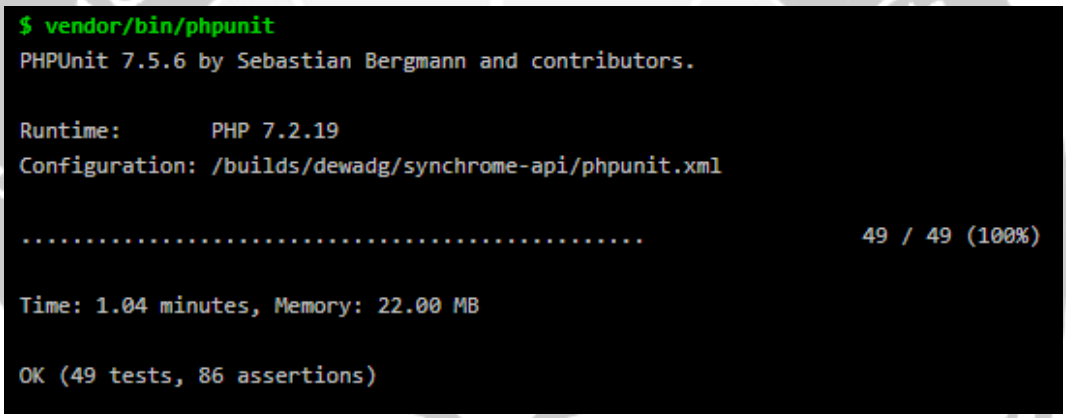
Contoh pengujian berikutnya adalah pengujian *method* create() pada kelas *AsnService*. Penulisan *test*-nya adalah sebagai berikut:

```
public function testCreate()
{
   $mocked_id = $this->faker->name;
   $mocked_agency_id = '1.1.01';
   $mocked_rank_id = '1a';
   $mocked_echelon_id = '1.1.01';
   $mocked_tpp_id = 1;
   $mocked_workshift_id = 1;
   $mocked_calendar_id = 1;
```

```
    $mocked_pin = $this->faker->name;
    $mocked_name = $this->faker->name;
    $mocked_phone = $this->faker->e164PhoneNumber;
    $mocked_address = $this->faker->streetAddress;

    $asn = $this->test_asn_service->create([
        'id' => $mocked_id,
        'agency_id' => $mocked_agency_id,
        'rank_id' => $mocked_rank_id,
        'echelon_id' => $mocked_echelon_id,
        'tpp_id' => $mocked_tpp_id,
        'workshift_id' => $mocked_workshift_id,
        'calendar_id' => $mocked_calendar_id,
        'pin' => $mocked_pin,
        'name' => $mocked_name,
        'phone' => $mocked_phone,
        'address' => $mocked_address,
    ]);

    $this->assertInstanceOf(Asn::class, $asn);
    $this->assertEquals($mocked_id, $asn->id);
    $this->assertEquals($mocked_agency_id, $asn->agency_id);
    $this->assertEquals($mocked_rank_id, $asn->rank_id);
    $this->assertEquals($mocked_echelon_id, $asn->echelon_id);
    $this->assertEquals($mocked_tpp_id, $asn->tpp_id);
    $this->assertEquals($mocked_workshift_id, $asn->workshift_id);
    $this->assertEquals($mocked_calendar_id, $asn->calendar_id);
    $this->assertEquals($mocked_pin, $asn->pin);
    $this->assertEquals($mocked_name, $asn->name);
    $this->assertEquals($mocked_phone, $asn->phone);
    $this->assertEquals($mocked_address, $asn->address);
}
```

Dalam potongan kode di atas, pengujian dimulai dengan mempersiapkan data ekspektasi untuk ASN yang akan didaftarkan. Kemudian data-data tersebut diinputkan ke *method* create() dan keluarannya akan diuji. Pengujian pertama adalah apakah benar keluarannya merupakan obyek dari *model* Asn, kemudian dilanjutkan dengan pengecekkan hasil data-data yang disimpan apakah sudah cocok dengan data yang awal diinputkan. Apabila salah satu dari pengujian tersebut gagal, maka *test* ini dinyatakan gagal.

Seluruh *test* akan dijalankan dan PHPUnit akan melaporkan hasil pengujian. Berdasarkan *test* yang ditulis pada *service layer*, berikut adalah hasil pengujiannya:



Gambar 5.4 – Hasil Pengujian *Synchrome*

Gambar 5.4 menunjukkan hasil pengujian pada tiap *service layer Synchrome*. Dari total 49 *test* dan 86 *assertion* seluruhnya berhasil. Berdasarkan hasil ini, *Synchrome* dinyatakan sudah memenuhi kriteria untuk di-*deploy*.

5.4 Analisis Kelebihan dan Kekurangan Sistem

Kelebihan yang dimiliki oleh *Synchrome* yang telah dirancang ini antara lain:

1. Dapat mengakomodir beberapa OPD dan mengakses laporan secara terpusat.
2. Proses merekaman log kehadiran tidak memerlukan koneksi internet secara terus menerus. Internet hanya diperlukan ketika akan menyinkronisasikan data ke *server* pusat.

3. Aplikasi *Synchrome Desktop* bersifat *cross-platform* yang berarti dapat diinstalasikan baik pada sistem operasi Windows, Linux, maupun macOS.

Serta tidak luput juga pengembangan *Synchrome* ini dari kekurangan seperti:

1. Konektivitas ke mesin pemindai sidik jari sangat bergantung pada spesifikasi komputer dan perangkat keras. Sangat rentang terhadap *down*.
2. Belum dapat mengakomodir seluruh jenis absensi pada ASN.
3. Belum memiliki fitur untuk sinkronisasi data secara luring.
4. Belum bisa mengakomodir mutasi ASN.

Pada bab implementasi dan pengujian perangkat lunak ini telah dijelaskan mengenai definisi sistem, implementasi sistem, dan hasil pengujian sistem. Pada bab selanjutnya yaitu penutup, akan diberikan kesimpulan dan saran yang didapatkan selama pembuatan tugas akhir ini.