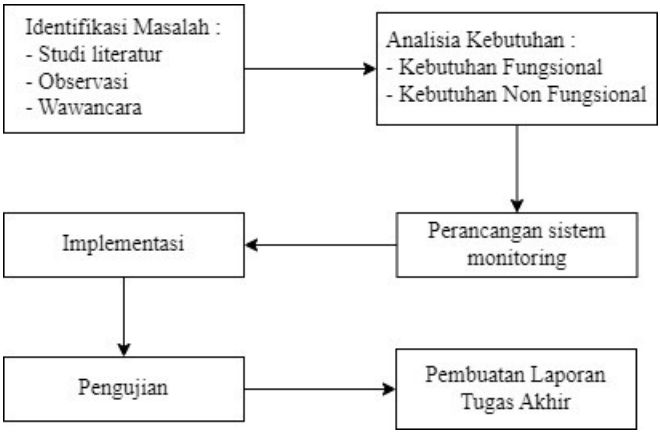
BAB III
METODOLOGI PENELITIAN

3.1 Alur Penelitian

Penelitian tentang Rancang Bangun Sistem Monitoring Kinerja Jaringan Berbasis Web Pada Zaha.net ini akan dibangun menggunakan metode pengembangan sistem yang terdiri dari langkah-langkah yang dimodelkan dalam bentuk diagram alur penelitian. Alur penelitian digunakan sebagai acuan atau pedoman dalam agenda penelitian yang akan dilakukan agar penelitian dapat berjalan secara terstruktur dan dapat diselesaikan tepat pada waktunya, serta agar penelitian dapat berjalan sesuai dengan yang diharapkan. Adapun metode yang digunakan yaitu dengan studi kasus dengan langkah-langkah sebagai berikut :



Gambar 3.1 Alur penelitian

Berdasrkan Gambar 3.1 dapat di lihat proses penelitian diawali dengan mengidentifikasi masalah melalui kajian literatur, observasi, dan wawancara. Analisis kebutuhan

dilakukan setelah identifikasi sebelumnya. Analisis kebutuhan mengacu pada identifikasi masalah dimana wawancara dilakukan untuk mengumpulkan informasi sebagai dasar proses monitoring dilakukan. Setelah dilakukan analisis kebutuhan, dibuatlah rancangan proses monitoring dengan diagram alir, kemudian diimplementasikan menggunakan protokol SNMP. Untuk memastikan sistem yang dibuat sesuai dengan yang diharapkan, maka langkah terakhir adalah melakukan pengujian sistem. Ketika sistem monitoring berhasil dibuat, maka laporan tugas akhir akan dibuat.

3.2 Identifikasi Masalah

Identifikasi masalah yang dilakukan dalam penelitian ini menggunakan studi literatur yang berupa jurnal, buku dan juga artikel yang terkait dengan penelitian ini sebagai sumber data sekunder. Observasi dan wawancara juga dilakukan kepada responden yang merupakan pengelola badan usaha Zaha.Net.

3.2.1 Studi Pustaka

Metode studi pustaka dilakukan dengan mencari referensi seputar yang mendukung penelitian yang memberikan informasi yang memadai seputar Zaha.net yang meliputi beberapa teori yang diambil melalui jurnal yang berhubungan.

3.2.2 Wawancara

Wawancara merupakan suatu proses komunikasi yang melibatkan dua pihak atau lebih, di mana satu pewawancara mengajukan pertanyaan kepada pihak

lainnya dengan tujuan untuk mengumpulkan informasi, pendapat, atau data tertentu yang akurat.

Beberapa poin pertanyaan yang penulis ajukan saat wawancara antara lain :

1. Mengapa Zaha.net membutuhkan sistem monitoring kinerja jaringan?
   Jawaban : Zaha.net membutuhkan sistem monitoring kinerja jaringan untuk memastikan bahwa layanan internet yang disediakan kepada pelanggan selalu dalam kondisi optimal. Dengan sistem monitoring, kami dapat mendeteksi masalah jaringan sejak dini, sebelum berdampak luas pada pelanggan. Hal ini memungkinkan kami untuk mengambil tindakan lebih cepat, meminimalkan *downtime*, dan meningkatkan kepuasan pelanggan melalui layanan yang lebih handal.
2. Apa resiko yang dapat dihindari dengan adanya sistem monitoring jaringan?
   Jawaban : Dengan adanya sistem monitoring jaringan, Zaha.net dapat menghindari berbagai risiko seperti waktu henti yang berkepanjangan yang dapat merugikan pelanggan dan mencegah reputasi Zaha.net rusak akibat gangguan layanan yang sering terjadi.

Dalam laporan ini, penulis hanya menyertakan dua poin pertanyaan hasil wawancara yang relevan dengan usulan penulis untuk membuat sistem monitoring jaringan di Zaha.net.

3.2.3 Observasi

Studi lapangan atau observasi merupakan aktivitas terhadap suatu proses atau objek dengan

maksud merasakan dan kemudian memahami pengetahuan dari sebuah fenomena berdasarkan pengetahuan dan gagasan yang sudah diketahui sebelumnya, untuk mendapatkan informasi yang dibutuhkan untuk melanjutkan suatu penelitian.

Adapun permasalahan yang di dapat dari hasil identifikasi masalah pada Zaha.net yaitu adanya kesulitan dalam memantau kondisi jaringan secara *real-time*, sehingga sulit bagi teknisi untuk melakukan tindakan secara cepat. Keterlambatan dalam mendeteksi dan menangani masalah jaringan dapat berdampak negatif pada kualitas layanan yang diberikan kepada pelanggan.

3.3 Analisis Kebutuhan

Pada tahap ini, analisis menentukan sebuah topik yang akan dijadikan sebagai pokok permasalahan, dari setiap pokok permasalahan yang ada akan dijadikan acuan untuk menentukan tujuan Rancang Bangun Sistem Monitoring Kinerja Jaringan Berbasis Web Pada Zaha.net. Pada tahap ini juga diadakan analisis kebutuhan untuk menentukan apa saja yang menjadi kebutuhan dalam sistem monitoring sebagai rencana pengembangan Zaha.net. Analisis kebutuhan pada penelitian dibagi menjadi dua, kebutuhan fungsional dan kebutuhan non-fungsional.

3.3.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan apa yang harus dilakukan oleh sistem. Ini mencakup fitur dan fungsi yang harus disediakan oleh sistem untuk

memenuhi kebutuhan pengguna. Berikut adalah analisis kebutuhan fungsional dari monitoring konektivitas jaringan di Zaha.net :

- Kemampuan melakukan pemantauan secara *real-time* terhadap kondisi jaringan
- Pemberitahuan kondisi jaringan secara konsisten dan tepat waktu
- Melihat status koneksi
- Menghapus *users* atau perangkat
- Melihat riwayat kondisi jaringan atau *log* aktivitas

3.3.2 Kebutuhan non-fungsional

Kebutuhan ini merujuk pada aspek-aspek yang bukan fitur atau perilaku fungsional suatu objek penelitian, melainkan karakteristik-karakteristik lain yang penting untuk keberhasilan dan kinerja objek penelitian secara keseluruhan. Kategorinya mencakup berbagai aspek, seperti performa, keamanan, keandalan, skalabilitas, dan lainnya. Berikut adalah analisis kebutuhan non-fungsional dari monitoring konektivitas jaringan di Zaha.net :

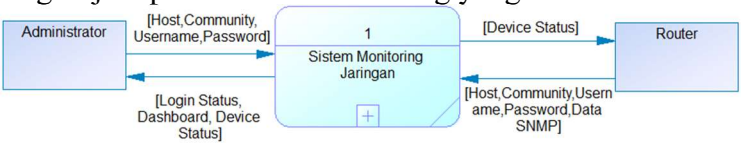
- Sistem monitoring disertai dengan *username* dan *password*
- Sistem moinitoring dapat menyimpan riwayat konektivitas jaringan
- Sistem monitoring membutuhkan IP *address* perangkat yang terkoneksi

3.4 Perancangan Sistem

Setelah tahapan analisis kebutuhan sistem, maka langkah selanjutnya adalah tahapan perancangan sistem. Tahap ini memiliki peran penting karena akan menjadi fondasi untuk pengembangan dan implementasi sistem monitoring yang dibangun.

3.4.1 Diagram Konteks

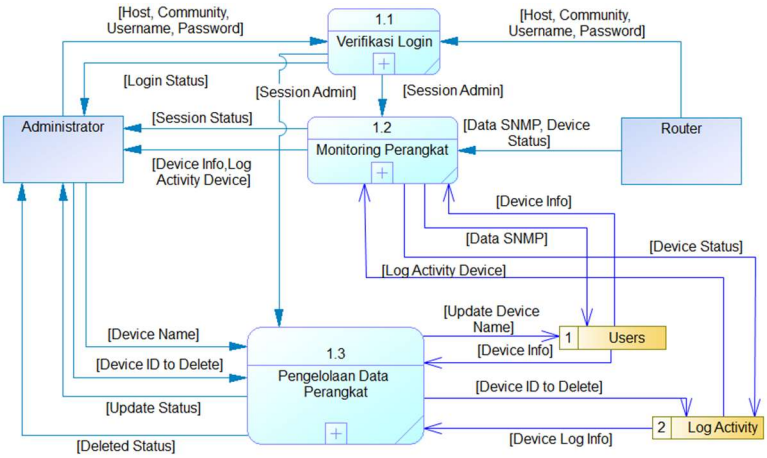
Diagram ini menampilkan sistem sebagai satu entitas tunggal dan memperlihatkan semua interaksi antara yang berhubungan dengannya. Diagram konteks membantu dalam memahami batasan sistem dan bagaimana sistem berinteraksi dengan komponen eksternal. Gambar 3.2 menunjukaan interaksi antar entitas yang terjadi pada sistem monitoring yang dibuat.



Gambar 3.2 Diagram Konteks

3.4.2 DFD Level 1

DFD level 1 adalah gambaran umum sistem monitoring jaringan yang memiliki 2 entitas yaitu entitas admininstrator dan entitas *router*. Kemudian memiliki proses yaitu, proses *login*, monitoring perangkat, dan kelola data perangkat. Proses tersebut dapat dibuktikan pada Gambar 3.3.



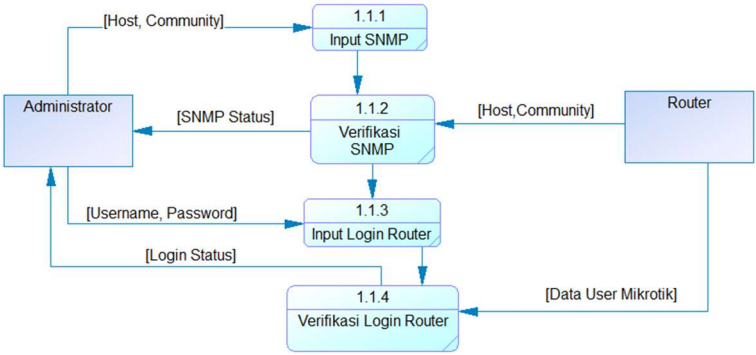
Gambar 3.3 DFD level 1

3.4.3 DFD Level 2

Karena proses utama dipecah menjadi beberapa subsistem, masing-masing dengan peran yang berbeda, DFD Level 2 menjelaskan aliran sistem secara lebih mendalam dan secara keseluruhan. Gambaran yang lebih lengkap tentang proses-proses yang ada dalam suatu sistem informasi disediakan oleh DFD Level 2.

3.4.3.1 DFD Level 2 Proses *Login*

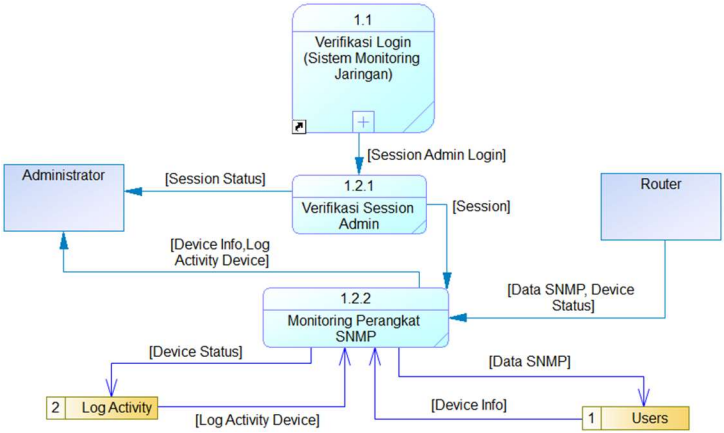
Pada Gambar 3.4 DFD level 2 proses *login* dimana pada proses ini dilakukan oleh administrator. Pada proses *login* ini administrator perlu memiliki akses terhadap infrormasi data perangkat berupa *hostname*, *community*, *username* & *password* untuk bisa mengakses halaman *website*. Kemudian dari informasi tersebut sistem akan memverifikasi apakah identitas yang di *input* valid atau tidak.



Gambar 3.4 DFD level 2 *login*

3.4.3.2 DFD Level 2 Proses Monitoring Perangkat

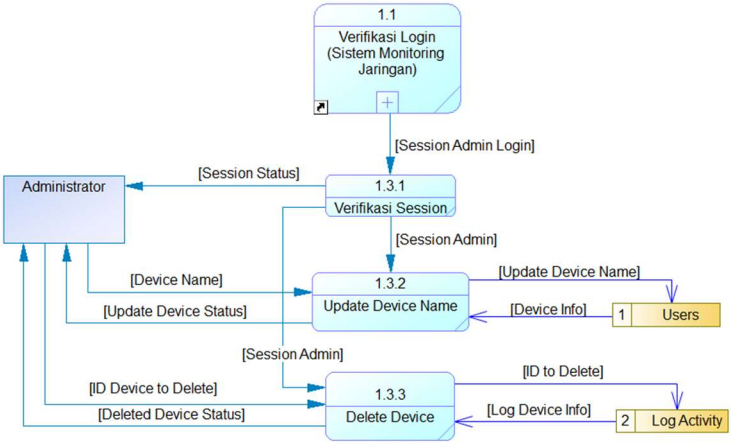
Pada Gambar 3.5 DFD level 2 proses monitoring perangkat, memberikan detail lebih rinci mengenai bagaimana data dikumpulkan, disimpan, dan dilaporkan dalam sistem monitoring perangkat. Pada proses ini administrator masuk, kemudian sistem akan memverifikasi sesi *login* dan proses monitoring dapat dilakukan.



Gambar 3.5 DFD level 2 monitoring perangkat

3.4.3.3 DFD Level 2 Proses Pengelolaan Data Perangkat

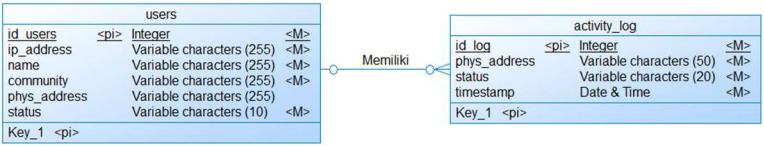
Gambar 3.6 DFD level 2 proses kelola data perangkat menunjukkan proses bagaimana data perangkat dikelola, yaitu bagaimana data perangkat diubah (*edit*) dan dihapus (*delete*).



Gambar 3.6 DFD level 2 Pengelolaan data perangkat

3.4.4 *Conceptual Data Model* (CDM)

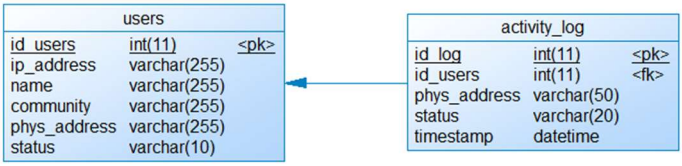
Model data fisik, yang pada akhirnya akan dibuat dalam bentuk model data konseptual (CDM), adalah versi pertama dari konsep desain basis data. Desain CDM ini menggambarkan entitas, atribut, dan hubungan di antara entitas-entitas tersebut. Interaksi antar entitas seperti pada Gambar 3.7.



Gambar 3.7 CDM

3.4.5 *Physical Data Model* (PDM)

Model data fisik (PDM) membantu menentukan struktur rinci elemen data dalam sistem dan hubungan antara elemen data. PDM adalah representasi yang lebih terperinci dan spesifik dari struktur basis data, menunjukkan bagaimana data disimpan secara fisik dalam *database*. Ini mencakup detail implementasi seperti tabel, kolom, indeks, tipe data, dan *constraint*. Interaksi tersebut seperti pada Gambar 3.8.



Gambar 3.8 PDM

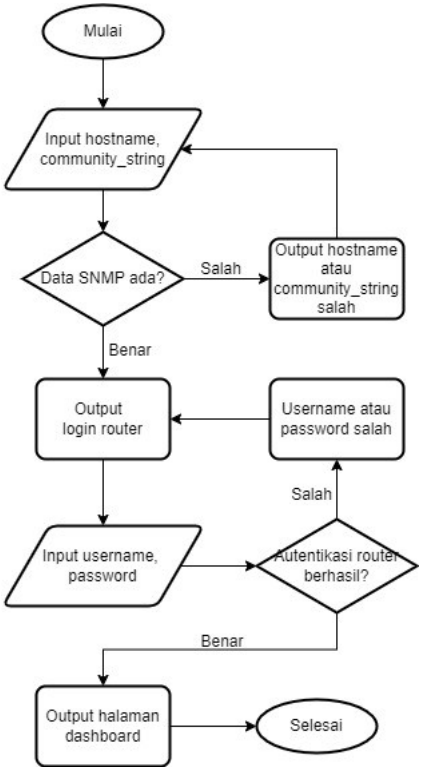
3.5 *Flowchart*

Flowchart menyediakan gambaran yang lebih mendetail tentang langkah-langkah spesifik yang diambil dalam proses tertentu. Setelah DFD level 2 memberikan pandangan tentang aliran data, *flowchart* memecah proses-proses tersebut menjadi tindakan-tindakan yang lebih terperinci dan spesifik.

3.5.1 *Flowchart Login*

Untuk memulai *flowchart* terdapat *terminal* mulai untuk data *input hostname* dan *community_string*, lalu berlanjut ke *decision* untuk memverifikasi data SNMP ada, jika salah akan kembali ke *input hostname* dan *community_string*, jika benar menuju ke halaman *login router*, kemudian dilakukan *input username* dan *password,* kemudian berlanjut ke *decision* untuk autentikasi apakah

username dan *password* sudah sesuai, jika salah kembali ke *input username* dan *password*, jika benar menuju halaman dashboard. Gambar 3.9 menunjukkan bentuk *flowchart login*.



Gambar 3.9 *Flowchart login*

Source code di bawah adalah implementasi *flowchart* pada gambar 3.9.

```
public function login() {
    $data['title'] = 'Login Page';
    $this->load->view('login', $data);
}
```

```
    public function process_snmp() {
        $hostname = $this->input->post('hostname');
                        $community_string = $this->input->post('community_string');
                        $session = $this->system_model->get_snmp_data($hostname, $community_string);

        if ($session) {
            $oid = "1.3.6.1.2.1.1.1.0"; // OID untuk sysDescr
            $value = $session->get($oid);
            if ($value) {
                $confirm = $value;
                $data = [
                    'hostname' => $hostname,
                    'community_string' => $community_string,
                    'title' => 'Router Login',
                    'confirm' => $confirm
                ];
                $this->load->view('login', $data);
            }else{
                            $this->set_error_flash('Hostname atau Community Salah!');
                redirect('admin/login');
            }

        } else {
                $this->session->set_flashdata('error', 'Hostname atau Community String tidak valid.');
            redirect('admin/login');
```

```
        }
    }

    public function authenticate() {
        $hostname = $this->input->post('hostname');
                    $community_string = $this->input->post('community_string');
        $username = $this->input->post('username');
        $password = $this->input->post('password');

                                if ($this->system_model->authenticate_mikrotik($hostname, $username, $password)) {
            $this->session->set_userdata([
                'hostname' => $hostname,
                'username' => $username,
                'password' => $password,
                'community_string' => $community_string,
                'logged_in' => true
            ]);
            redirect('admin/dashboard');
        } else {
                $this->set_error_flash('Username atau password Router salah!');
            redirect('admin/login');
        }
    }
```

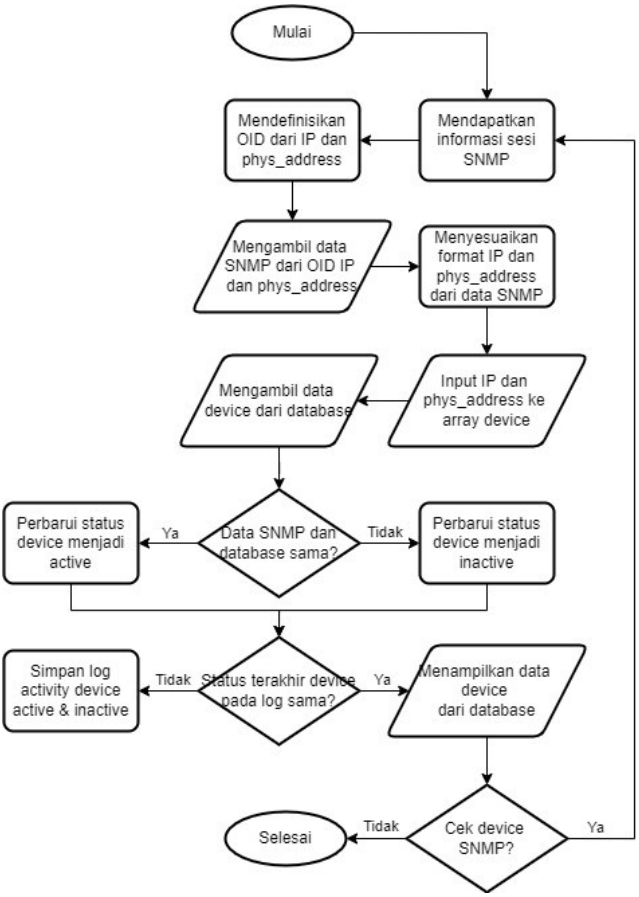
Pertama, fungsi *login()* menampilkan halaman *login*. Saat pengguna mengirimkan *form login*, fungsi

process_snmp() mengambil *hostname* dan *community_string* dari *input* POST dan mencoba mendapatkan data SNMP menggunakan model *system_model*. Jika sesi SNMP berhasil, fungsi ini mengambil deskripsi sistem (sysDescr) dari OID tertentu dan menampilkan konfirmasi pada halaman *login*. Jika gagal, fungsi ini mengatur pesan *error* dan mengarahkan kembali ke halaman *login*. Selanjutnya, fungsi *authenticate()* mengambil *hostname*, *community_string*, *username*, dan *password* dari *input* POST dan mencoba mengotentikasi pengguna ke *router* menggunakan model *system_model*. Jika otentikasi berhasil, data pengguna disimpan dalam sesi, dan administrator diarahkan ke halaman *dashboard*. Jika gagal, pesan *error* ditampilkan dan pengguna diarahkan kembali ke halaman *login*.

3.5.2 *Flowchart View Status*

Untuk memulai *flowchart* terdapat *terminal* mulai lalu berlanjut ke proses mendapatkan informasi sesi SNMP, setelah itu berlanjut ke proses mendefinisikan OID dari IP dan *phys_address*, kemudian mengambil data SNMP dari OID IP dan *phys_address*, selanjutnya menyesuaikan format IP dan *phys_address* dari data SNMP, proses selanjutnya *input* IP dan *phys_address array device*, kemudian mengambil data *device* dari *database*, lalu *decision* untuk data SNMP dari *database* sama, jika “ya” status *device* di perbarui menjadi *active*, jika “tidak” status *device* di perbarui menjadi *inactive*, berlanjut ke *decision* status terakhir *device* pada *log* sama, jika “tidak” simpan *log activity device active* & *inactive*,

jika “ya” menampilkan data *device* dari database, lanjut ke *decision* cek *device* SNMP, jika “tidak” maka selesai, jika “ya” maka kembali ke proses mendapatkan informasi sesi SNMP. Gambar 3.10 menunjukkan bentuk *flowchart view status*.



Gambar 3.10 *Flowchart view status*

Source code di bawah adalah implementasi *flowchart* pada gambar 3.10.

```
public function view_status() {
    $hostname = $this->session->userdata('hostname');
                $community_string = $this->session->userdata('community_string');

    if (!extension_loaded('snmp')) {
        echo json_encode(['error' => 'SNMP extension not loaded.']);
        return;
    }

        $session = new SNMP(SNMP::VERSION_2C, $hostname, $community_string, 500000, 2);

    $ip_address_oid = '1.3.6.1.2.1.4.22.1.3';
    $phys_address_oid = '1.3.6.1.2.1.4.22.1.2';

                $ip_addresses_raw = $session->walk($ip_address_oid);
                $phys_addresses_raw = $session->walk($phys_address_oid);

    $devices = [];

    foreach ($ip_addresses_raw as $key => $ip_address) {
        $ip_address_clean = str_replace('IpAddress: ', '', $ip_address);
        $devices[$ip_address_clean] = [
            'ip_address' => $ip_address_clean,
```

```
            'phys_address' => 'Unknown',
            'status' => 'active'
        ];
    }

        foreach ($phys_addresses_raw as $key => $phys_address) {
        $phys_address_clean = str_replace('Hex-STRING: ', '', $phys_address);
                    $phys_address_clean = implode(':', str_split($phys_address_clean, 2));

        // Memisahkan bagian kunci IP address dari $key
        $key_parts = explode('.', $key);
                    $ip_address_key = implode('.', array_slice($key_parts, -4));

        // Mengecek apakah $ip_address_key sudah ada di $devices
        if (isset($devices[$ip_address_key])) {
            // Jika sudah ada, update $phys_address
                $devices[$ip_address_key]['phys_address'] = $phys_address_clean;
        } else {
                // Jika belum ada, tambahkan entri baru ke $devices
            $devices[$ip_address_key] = [
                'ip_address' => $ip_address_key,
                'phys_address' => $phys_address_clean,
                'status' => 'active'
```

```
            ];
        }
    }

            $db_devices = $this->userdevice_model->get_all_devices();

    foreach ($db_devices as &$db_device) {
        $phys_address = $db_device['phys_address'];
        $ip_address = $db_device['ip_address'];

        if (isset($devices[$ip_address])) {
                    $this->db->where('phys_address', $phys_address);
            $this->db->update('users', [
                                'ip_address' => $devices[$ip_address]['ip_address'],
                'status' => 'active'
            ]);
                                $this->logactivity_model->log_activity($phys_address, 'active');
            unset($devices[$ip_address]);
        } else {
            $new_status = 'inactive';
            if ($db_device['status'] != $new_status) {
                    $this->db->where('phys_address', $phys_address);
                $this->db->update('users', [
                    'status' => $new_status
                ]);
```

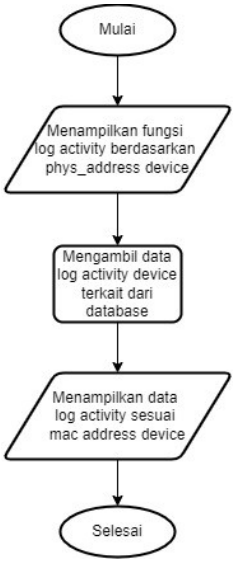
```
                                        $this->logactivity_model-
>update_log($phys_address, $new_status);
                }
            }
        }

        foreach ($devices as $device) {
                                        $this->db->where('ip_address',
$device['ip_address']);
            $query = $this->db->get('users');
            if ($query->num_rows() > 0) {
                                        $this->db->where('ip_address',
$device['ip_address']);
                $this->db->update('users', [
                    'phys_address' => $device['phys_address'],
                    'status' => 'active'
                ]);
            } else {
                $this->db->insert('users', [
                    'name' => '-',
                    'ip_address' => $device['ip_address'],
                    'phys_address' => $device['phys_address'],
                    'status' => 'active',
                    'community' => $community_string
                ]);
                                        $this->logactivity_model-
>update_log($device['phys_address'], 'active');
            }
        }
```

```
    echo json_encode(['devices' => $db_devices]);
}
```

Pertama, program mengambil *hostname* dan *community_string* dari sesi pengguna, serta memeriksa apakah ekstensi SNMP terpasang. Jika ekstensi tidak terpasang, program mengembalikan pesan kesalahan. Program membuat sesi SNMP dengan parameter tertentu dan menggunakan OID untuk mendapatkan alamat IP dan fisik (*phys_address*) dari perangkat. Data yang diperoleh kemudian diolah, alamat IP dibersihkan dan dimasukkan ke dalam *array $devices* dengan status '*active'*, sedangkan alamat fisik dikonversi ke format standar *phys_address* dan ditambahkan atau diperbarui di *array $devices*. Program kemudian mendapatkan semua perangkat dari *database* dan memperbarui status mereka berdasarkan data SNMP: perangkat yang ada dalam kedua sumber diatur ke '*active*', sementara perangkat yang hanya ada di *database* diatur ke '*inactive*'. Terakhir, perangkat baru dari data SNMP yang belum ada di *database* ditambahkan, dan *log activity* diperbarui.

3.5.3 *Flowchart Log Activity*

Untuk memulai *flowchart* terdapat *terminal* mulai lalu berlanjut menampilkan fungsi *log activity* berdasarkan *phys_address*, lalu berlanjut ke proses mengambil data *log activity device* terkait dari *database*, kemudian menampilkan data *log activity* sesuai *phys_address device*. Gambar 3.11 menunjukkan bentuk *flowchart log activity.*



Gambar 3.11 *Flowchart Log Activity*

Source code di bawah adalah implementasi *flowchart* pada gambar 3.11.

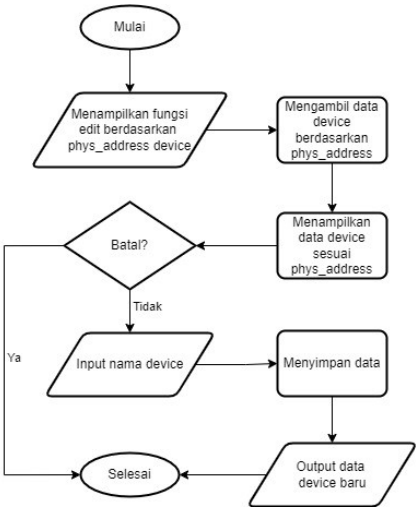
```
public function view_device_logs() {
    $phys_address = $this->input->post('phys_address');
                $logs = $this->logactivity_model->get_device_log($phys_address);
     echo json_encode(['logs' => array_slice($logs, -6)]); // Ambil 6 log terakhir
  }
```

Pertama, fungsi *view_device_logs()* menerima permintaan HTTP POST yang berisi alamat fisik perangkat (*phys_address*). Nilai ini kemudian diambil menggunakan *$this->input->post('phys_address')*.

Selanjutnya, fungsi ini menggunakan model *logactivity_model* untuk memanggil metode *get_device_log* dengan parameter alamat fisik perangkat tersebut untuk mengambil semua *log* aktivitas yang terkait dari *database*. Setelah *log* diambil, fungsi ini memotong *array log* tersebut untuk hanya mengambil 6 *log* terbaru menggunakan *array_slice($logs*, -6). Terakhir, fungsi ini mengubah *array log* yang sudah dipotong menjadi format JSON dengan *json_encode* dan mengirimkannya sebagai respons HTTP menggunakan *echo*. Proses ini memungkinkan pengguna untuk mendapatkan 6 *log* aktivitas terbaru dari perangkat.

3.5.4 *Flowchart Edit*

Untuk memulai *flowchart* terdapat *terminal* mulai lalu berlanjut menampilkan fungsi *edit* berdasarkan *phys_address device*, lalu berlanjut ke proses mengambil data *device* berdasarkan *phys_address*, kemudian menampilkan data *device* sesuai *phys_address*, lalu berlanjut ke *decision* batal, jika "ya" maka selesai, jika "tidak" maka *input* nama *device*, kemudian lanjut ke proses menyimpan data, dan kemudian *output*-nya data *device* baru. Gambar 3.12 menunjukkan bentuk *flowchart log activity.*



Gambar 3.12 *Flowchart edit*

Source code di bawah adalah implementasi *flowchart* pada gambar 3.12.

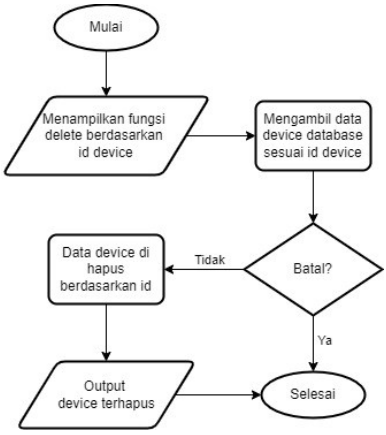
```
public function assign_name() {
      $id = $this->input->post('phys_address');
      $this->db->where('phys_address', $id);
       $this->db->update('users', ['name' => $this->input->post('name')]);
      echo json_encode(['success' => true]);
      redirect('admin/users');
   }
```

Ketika fungsi ini dipanggil, pertama-tama, akan mengambil nilai *phys_address* dari *input* POST yang dikirim oleh pengguna dan menyimpannya dalam variabel $id. Selanjutnya, fungsi ini membuat *query database* untuk mencari baris di tabel *users* yang memiliki *phys_address*

sesuai dengan nilai $id tersebut. Setelah itu, ia memperbarui kolom *name* pada baris yang ditemukan dengan nilai yang diterima dari *input* POST dengan *key name*. Setelah operasi *update* berhasil, administrator akan dialihkan ke halaman users melalui fungsi *redirect*. Dengan demikian, fungsi ini bertanggung jawab untuk mengubah nama pengguna berdasarkan alamat fisik yang diberikan dan memastikan pengguna kembali ke halaman manajemen pengguna setelah operasi selesai.

3.5.5 *Flowchart Delete*

Untuk memulai *flowchart* terdapat *terminal* mulai lalu berlanjut menampilkan fungsi *delete* berdasarkan *phys_address device*, kemudian lanjut ke proses mengambil data *device database* sesuai id *device*, lalu lanjut ke *decision* batal, jika "ya" maka selesai, jika "tidak" maka data *device* di hapus berdasarkan id, kemudian *output device* terhapus. Gambar 3.13 menunjukkan bentuk *flowchart log activity.*



Gambar 3.13 *Flowchart delete*

Source code di bawah adalah implementasi *flowchart* pada gambar 3.13.

```
public function delete() {
    $id = $this->input->post('id');

    // Ambil phys_address dari tabel users
    $this->db->select('phys_address');
    $this->db->from('users');
    $this->db->where('id', $id);
    $query = $this->db->get();
    $phys_address = $query->row()->phys_address;

        // Hapus dari tabel log_activity berdasarkan
phys_address
    $this->db->where('phys_address', $phys_address);
    $this->db->delete('activity_log');

    // Hapus dari tabel users
    $this->db->where('id', $id);
    $this->db->delete('users');

    echo json_encode(['status' => 'success']);
  }
```

Fungsi di atas dimulai dengan mengambil ID dari request POST, kemudian mencari alamat fisik (*phys_address*) yang terkait dengan ID tersebut di tabel users. Setelah menemukan *phys_address*, fungsi ini menghapus semua catatan di tabel *activity_log* yang

memiliki *phys_address* tersebut. Selanjutnya, fungsi menghapus baris dari tabel *users* yang sesuai dengan ID yang diberikan.

3.5 Implementasi

Berikut adalah persyaratan implementasi : VirtualBox dan OS antara lain *router* Mikrotik dan Windows XP sebagai sistem operasinya, bahasa pemrograman PHP, *framework CodeIgniter* dan *database* berbasis SQL untuk aplikasi monitoring pada *host*.