TUGAS AKHIR – K141502

RANCANG BANGUN APLIKASI PEMANTAUAN *WEB SERVICE* BERBASIS PROTOKOL REST DAN SOAP

ALFITRAH NURRAMADHAN SUDIRMAN
NRP 05111440000037

Dosen Pembimbing
Rizky Januar Akbar, S.Kom, M.Eng.
Royyana Muslim Ijtihadie, S.kom., M.Kom., PhD.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI1502

RANCANG BANGUN APLIKASI PEMANTAUAN *WEB SERVICE* BERBASIS PROTOKOL REST DAN SOAP

ALFITRAH NURRAMADHAN SUDIRMAN
NRP 05111440000037

Dosen Pembimbing
Rizky Januar Akbar, S.Kom., M.Eng.
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI1502

DESIGN AND IMPLEMENTATION OF REST AND SOAP BASED WEB SERVICE MONITORING APPLICATION

ALFITRAH NURRAMADHAN SUDIRMAN
NRP 05111440000037

Supervisors
Rizky Januar Akbar, S.Kom., M.Eng.
ROYYANA MUSLIM IJTIHADIE, S.Kom., M.Kom., PhD.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI RANCANG BANGUN APLIKASI PEMANTAUAN *WEB SERVICE* BERBASIS PROTOKOL REST DAN SOAP

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
Alfitrah Nurramadhan Sudirman
NRP: 05111440000037

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Rizky Januar Akbar, S.Kom., M.Eng.
NIP: 19870103 201404 1 001 ..... (pembimbing 1)

Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.
NIP: 19770824 200604 1 001 ..... (pembimbing 2)

SURABAYA
JULI 2018

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN APLIKASI PEMANTAUAN *WEB SERVICE* BERBASIS PROTOKOL REST DAN SOAP

Nama Mahasiswa	:	Alfitrah Nurramadhan Sudirman
NRP	:	05111440000037
Jurusan	:	Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	:	Rizky Januar Akbar, S.Kom., M.Eng.
Dosen Pembimbing 2	:	Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

ABSTRAK

Untuk mempercepat recovery apabila terjadi kegagalan pada sebuah sistem informasi, diperlukan pemantauan secara berkala pada endpoint-endpoint milik web service untuk mendeteksi apakah terdapat endpoint yang tidak bekerja dengan baik. Pengecekan secara manual pada endpoint-endpoint akan memperberat pekerjaan administrator sistem. Maka dari itu, diperlukan adanya sebuah aplikasi pemantauan web service yang dapat memantau kinerja endpoint-endpoint pada web service secara otomatis dan terjadwal.

Implementasi tugas akhir ini terdiri dari tiga bagian perangkat lunak, yaitu Server Uji, Server Pusat dan Aplikasi Dashboard. Bagian Server Uji berfungsi untuk mengeksekusi pengujian dengan kasus uji. Bagian Server Pusat berfungsi untuk memberi perintah pengiriman secara berkala pada Server Uji dengan fungsi Scheduler yang secara otomatis dapat mengirim perintah pengujian secara berkala pada Server Uji.kedua perangkat lunak ini diimplementasikan dengan Bahasa pemrograman java dan maven framework serta jersey web service restful library Aplikasi Dashboard berfungsi untuk melakukan pengaturan serta melihat hasil pengujian, dan diimplementasikan dengan menggunakan Bahasa pemrograman PHP dan framework

Codeigniter. Tipe HTTP protocol yang dapat digunakan pada aplikasi ini adalah: REST (GET dan POST) dan SOAP. Tipe otentikasi HTTP yang bisa digunakan pada aplikasi ini adalah: Basic, Digest dan Bearer.

Dari hasil pengujian fungsionalitas aplikasi ini, dapat disimpulkan bahwa aplikasi berjalan sesuai dengan tujuan dibuatnya aplikasi. Pengembangan lebih lanjut diperlukan agar fungsi yang ada pada aplikasi ini dapat bekerja dengan lebih maksimal.

Kata Kunci*: web service, pemantauan, endpoint, request, REST, SOAP, scheduler, pengetesan otomatis*

DESIGN AND IMPLEMENTATION OF REST AND SOAP-BASED WEB SERVICE MONITORING APPLICATION

Name	: Alfitrah Nurramadhan Sudirman
NRP	: 05111440000037
Jurusan	: Teknik Informatika FTIf-ITS
Supervisor I	: Rizky Januar Akbar, S.Kom., M.Eng.
Supervisor II	: Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

ABSTRACT

To ease the detection and recovery process in the case of system information service failures, it is necessary to have the endpoints in the system monitored time after time. Doing regular manual-way checking may turn out to be bothersome by the system administrators, if the system is big enough. To overcome this problem, an application which is able to do automated and scheduled web service monitoring is needed.

This final project implementation consists of three parts. Those three parts are Testing Servers, Main Server and a Dashboard application. Testing Server part executes the monitoring action. the Main Server sends orders to Testing Servers to execute the monitoring action, in automated and scheduled manner. These two parts are implemented with Java programming language and maven framework. the Dashboard Application shows the summary of all monitoring actions, and also capable of being used by system administrators to set configurations and settings of the monitoring actions. The Dashboard Application is implemented with PHP programming language and codeigniter framework. The HTTP Prortocols which is supported in this system are REST (GET and POST) and also SOAP. The HTTP Authorization

headers which is supported in this system are Basic, Digest and Bearer token authorization.

From some functionality testings that are conducted with this application, it can be concluded that this application can fulfill its functional requirements. Further Enhancements is needed to be done to this application to maximize its potential.

***Keywords**: Web service, Monitoring, Endpoint, Request, REST, SOAP, Scheduler, Automated Testing*

KATA PENGANTAR

Alhamdulillahirobbil‘alamiin, puji syukur kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul:

Rancang Bangun Aplikasi Pemantauan *Web Service* Berbasis Protokol REST dan SOAP

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Allah SWT atas segala nikmat dan rahmat yang telah diberikan selama ini,
2. Bapak Senthot Sudirman dan Ibu Nurul Ariningsih selaku orang tua penulis, Amalia Nuranissa Sudirman dan Adiyat Nurkautsar Sudirman selaku kakak penulis, serta keluarga besar penulis yang tiada henti-hentinya mencurahkan kasih sayang, perhatian dan doa kepada penulis selama ini,
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku dosen pembimbing I yang selalu memberikan motivasi dan membimbing penulis selama pengerjaan tugas akhir,
4. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD. selaku dosen pembimbing II yang telah memberikan nasihat, arahan, dan bantuan terutama pada bagian pengerjaan buku, sehingga penulis dapat menyelesaikan pengerjaan tugas akhir ini,
5. Bapak dan Ibu dosen Teknik Informatika ITS yang telah membina dan memberikan ilmu kepada penulis selama menempuh studi di Teknik Informatika ITS,
6. Teman-teman angkatan 2014 khususnya Dewangga Okta W. yang telah memberikan semangat dan motivasi, serta berbagi ilmu selama penulis berkuliah di Informatika ITS,
7. Sang Norma Lintang Asmara, yang selalu ada untuk

memberi semangat dan dukungan dari jauh kepada penulis,

8. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu persatu.

Bagaimanapun juga penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juli 2018

Alfitrah Nurramadhan Sudirman

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Pada era teknologi informasi, penggunaan sistem informasi sangatlah berpengaruh pada berbagai bidang karena sistem informasi dapat mempermudah pekerjaan manusia. Akan tetapi, dengan terus meningkatnya jumlah penduduk, pengguna sistem informasi yang ada akan terus bertambah. Adakalanya, karena satu dan lain hal, sistem informasi dapat mengalami gangguan dalam melayani penggunanya. Gangguan yang dapat muncul diantaranya adalah penurunan kecepatan akses pada suatu *web service*, respon *web service* yang tidak sesuai dengan seharusnya, dan kegagalan *web service* dalam menerima *request*.

Salah satu cara yang dapat dilakukan untuk mempercepat *recovery* pada sistem jika terjadi kegagalan adalah melakukan pengecekan secara berkala pada *endpoint-endpoint* milik *web service* yang terdapat pada sistem informasi yang ada untuk mendeteksi apakah terdapat *endpoint* pada *web service* yang tidak bekerja dengan baik. Terdapat dua kemungkinan metode pelaksanaan pengecekan *endpoint* pada sistem, yaitu secara manual dan otomatis. Pengecekan secara manual dapat dilakukan oleh *administrator* sistem dengan cara merekayasa sebuah kasus uji berupa *request* yang inputnya disesuaikan untuk sebuah *endpoint* dan mengirim kasus uji tersebut ke *server*, kemudian menganalisa respon yang dikembalikan oleh *server*. Respon yang dikembalikan oleh *server* dapat menunjukkan kondisi dari *endpoint* tersebut. Akan tetapi, metode pengecekan manual secara berkala cenderung memakan waktu yang lama. Selain itu, perubahan kinerja *endpoint* dari waktu ke waktu lebih sulit untuk

diamati. Maka dari itu, diperlukan adanya aplikasi pemantauan *web service* yang dapat melakukan pengecekan berkala fungsi *endpoint* pada *server* secara otomatis. Terdapat dua bagian dari aplikasi ini, yaitu *dashboard* pemantauan *web service* dan *request scheduler engine*. *Dashboard* pemantauan *web service* pada aplikasi ini berfungsi untuk memudahkan pengguna aplikasi dalam mengatur proses pengecekan otomatis yang dilakukan oleh aplikasi (mengatur kasus uji, mengatur interval pengujian, dan lain-lain), serta menampilkan informasi-informasi dan *summary* dari hasil pengecekan yang dilakukan oleh aplikasi. Sedangkan perangkat lunak *scheduler* dan *request executor* berfungsi untuk mengeksekusi proses pengecekan *endpoint* sesuai dengan pengaturan yang dibuat oleh pengguna aplikasi.

Adanya aplikasi untuk mengecek fungsi *endpoint* pada *web service* secara otomatis diharapkan dapat mempermudah *administrator* sistem dalam mengawasi kinerja sistem informasi dan mendeteksi masalah-masalah yang ada pada sistem sedini mungkin.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana implementasi pengujian kinerja *web service* dengan menggunakan kasus uji berbasis protokol REST atau SOAP pada aplikasi pemantauan *web service*?
2. Bagaimana implementasi *scheduler* pada aplikasi pemantauan *web service*?
3. Bagaimana implementasi bagian *dashboard* pemantauan *web service* pada aplikasi pemantauan *web service*?

1.3. Batasan Permasalahan

Beberapa batasan masalah yang menjadi batas pada tugas akhir ini adalah:

1. Pembuatan implementasi sistem menggunakan Bahasa pemrograman Java
2. Jenis protokol HTTP yang dapat diuji oleh sistem adalah Protokol REST dan SOAP
3. Jenis *request* pada Protokol REST yang dapat digunakan yaitu GET dan POST.
4. Jenis otentikasi HTTP yang dapat diterapkan pada *request* yaitu: *Basic authentication*, *Digest authentication*, dan *Bearer Token authentication*.
5. Seluruh parameter *request* yang ada hanya dapat berupa teks.

1.4. Tujuan

Tujuan dari tugas akhir ini adalah:

1. Menerapkan pengujian kinerja *web service* dengan menggunakan kasus uji *request* berbasis protokol REST atau SOAP pada aplikasi pemantauan *web service*.
2. Menerapkan pembuatan *scheduler* pada aplikasi pemantauan *web service*.
3. Menerapkan bagian *dashboard* pemantauan *web service* pada aplikasi pemantauan *web service.*

1.5. Manfaat

Aplikasi Pemantauan *Web Service* diharapkan dapat mempermudah pekerjaan *administrator* sistem informasi dalam melakukan pengecekan kinerja *web service* secara otomatis dan terjadwal. Selain itu, aplikasi ini juga dapat mempermudah pengujian koneksi *web service* dengan banyak *server* sekaligus.

1.6. Metodologi

Adapun beberapa tahap dalam proses pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Studi Literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi. Adapun tahap studi literatur dan pemahaman sistem adalah sebagai berikut.

- Mencari literatur dan mempelajari beberapa *library* pada bahasa Java yang dibutuhkan dalam membuat *request* berbasis protokol REST dan SOAP.
- Mempelajari tentang struktur *request* berbasis protokol REST dan SOAP.
- Mempelajari literatur mengenai pembuatan aplikasi Java *web service*.
- Mempelajari literatur mengenai *scheduler* pada bahasa pemrograman Java.
- Mempelajari konsep *client-server* yang akan diimplementasikan pada aplikasi, serta API yang terkait.

2. Analisis dan desain perangkat lunak

Tahap ini meliputi perumusan aktor, kebutuhan fungsional, kasus penggunaan, diagram aktivitas, dan diagram sekuens.

3. Implementasi

Implementasi yang dilakukan adalah rancang bangun aplikasi pemantauan *web service* berbasis *web* pada *server* pusat, serta *request engine* dan *scheduler engine* pada *server* uji dan *server* pusat.

Implementasi aplikasi ini dibangun dengan menggunakan bahasa pemrograman Java dengan menggunakan *IDE Eclipse Kepler R2* dan pengembangan aplikasi web menggunakan *server* XAMPP dan text editor Sublime Text 3.

Bahasa Pemrograman yang digunakan pada implementasi yaitu Maven dengan archetype Jersey versi 2.16 pada *Scheduler Engine* dan *Request Engine*, dan PHP pada Aplikasi *Dashboard*. Kerangka kerja yang digunakan pada implementasi yaitu Maven dengan archetype Jersey versi 2.16 pada *Scheduler Engine* dan *Request Engine*, dan CodeIgniter versi 3.1.7 pada Aplikasi *Dashboard*.

4. Uji Coba dan Evaluasi

Pada tahap ini, akan dilakukan uji coba pembuatan dan eksekusi kasus uji pada *web service* yang dieksekusi oleh *scheduler* dengan aplikasi yang telah dibuat. Beberapa hal yang akan diujicobakan adalah sebagai berikut.

1. Membuat kasus uji dengan jenis-jenis *request* yang ada.
2. Memasang kasus uji pada *server* uji.
3. Mengatur parameter pemasangan kasus uji pada *server* uji.
4. Menguji fungsionalitas *scheduler* milik aplikasi pada *server* pusat.
5. Mengeksekusi kasus uji pada *server-server*.
6. Menampilkan hasil pengujian pada kasus uji secara *real-time*.

5. Penyusunan Buku Tugas Akhir

Tahap ini merupakan tahap penyusunan laporan berupa buku tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh teori, implementasi,

serta hasil pengujian yang telah dikerjakan. Kesimpulan dan saran diberikan pada buku Tugas akhir ini sebagai tambahan terhadap aplikasi.

1.7. Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, buku Tugas Akhir ini diharapkan dapat berguna bagi pembaca yang berminat untuk melakukan pengembangan pada aplikasi yang telah dibuat. Secara garis besar, buku Tugas Akhir ini terdiri atas beberapa bagian, yaitu sebagai berikut.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Dasar Teori

Memaparkan hasil studi literatur yang digunakan menyelesaikan Tugas Akhir ini, terdiri atas, pustaka mengenai *scheduler*, *request* SOAP dan REST, HTTP *Header*, SQL Server, dan CodeIginter.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi rancangan sistem yang akan dibangun, dimulai dari deskripsi umum sistem, batasan sistem, kasus penggunaan aplikasi, dan rancangan antarmuka pengguna, dan rancangan basis data.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan pada bab sebelumnya serta implementasi bagian-bagian penunjang aplikasi termasuk implementasi sistem dan implementasi antarmuka pengguna.

Bab V Pengujian dan Evaluasi

Bab ini membahas tentang pengujian

fungsionalitas aplikasi dengan metode kotak hitam (*black box testing*) untuk mengetahui apakah implementasi aplikasi dapat menjalankan fungsi yang sesuai dengan tujuan dibuatnya aplikasi.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian pada bab sebelumnya. Pada bab ini juga dibahas saran-saran mengenai hal-hal yang dapat ditambahkan pada sistem sebagai perkembangan lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan dalam proses pengembangan aplikasi serta penulisan buku Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar kode yang ada pada aplikasi ini.

[Halaman ini sengaja dikosongkan]

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori serta literatur yang menjadi dasar dari pembuatan implementasi aplikasi pada Tugas Akhir ini. Teori-teori tersebut meliputi pengertian dan beberapa analisis terkait HTTP *request, Scheduler*, dan perangkat-perangkat pendukung pengembangan aplikasi.

2.1. HTTP

HTTP (*Hypertext Transfer Protocol*) adalah protokol dasar yang digunakan oleh *World Wide Web* dan protokol ini mendefinisikan bagaimana pesan diformat, dikirimkan, dan tindakan apa yang dilakukan oleh *server* dan *web browser* (klien) dalam menanggapi berbagai perintah [1]. HTTP berfungsi sebagai protokol *request-response* pada model klien-*server*. Sumber daya yang hendak diakses dengan HTTP diidentifikasi dengan menggunakan *Uniform Resource Identifier* (URI), atau lebih khusus menggunakan *Uniform Resource Locator* (URL), dengan skema URI http atau https.

Klien atau *web browser* mengirimkan pesan permintaan (*request*) HTTP ke *server*. *Server*, yang menyediakan sumber daya seperti file HTML dan konten lainnya, atau melakukan fungsi lain atas nama klien, mengembalikan pesan tanggapan (*response)* ke klien. *Response* berisi informasi status penyelesaian atas *request* dan mungkin juga berisi konten yang diminta dalam badan pesannya (*message body)*.

Terdapat empat metode *request* pada protokol HTTP yang umum ditemukan. Metode-metode tersebut adalah sebagai berikut. [2]

1. **GET**
   Mendapatkan (*read)* sebuah sumber daya (*resource*) yang diidentifikasi dengan URI (*Uniform Resource Identifier*).
2. **POST**

Mengirimkan sumber daya *(resource*) ke *server*. Digunakan untuk membuat (c*reate*) sumber daya baru.

3. **PUT**
   Mengirimkan sumber daya (r*esource*) ke *server*. Digunakan untuk memasukkan (*insert*) atau memperbarui *(update*) sumber daya yang tersimpan.
4. **DELETE**
   Menghapus (*delete)* sumber daya (*resource*) yang diidentifikasi dengan URI.

Metode HTTP yang akan digunakan pada tugas akhir dapat dipilih pada aplikasi pemantauan *web service* dan diterapkan pada *request scheduler engine*.

Pada HTTP terdapat beberapa macam *response*. Terdapat empat kategori *response* yang biasa dijumpai pada HTTP *response*. Kategori-kategori tersebut adalah sebagai berikut. [3]

1. 2xx, adalah kode respon yang menandakan bahwa *request* berhasil.
2. 3xx, adalah kode respon yang menandakan bahwa terdapat *redirect* pada *endpoint* teserbut.
3. 4xx, adalah kode respon yang menandakan bahwa *request* mengalami kesalahan pada sisi klien.
4. 5xx, adalah kode respon yang menandakan bahwa *request* mengalami kesalahan pada sisi *server*.

Kode respon yang muncul akan digunakan sebagai patokan dalam mengevaluasi kondisi *endpoint* pada aplikasi pemantauan *web service*.

2.2. PHP

PHP (*PHP: Hypertext Preprocessor*) adalah bahasa *scripting* pada sisi *server* yang umumnya digunakan untuk pengembangan web [4]. Kode PHP dapat disematkan ke *markup* HTML atau HTML5, atau dapat digunakan dalam kombinasi dengan berbagai

sistem *template* web, sistem pengelolaan konten web (*web content managemen system)* dan kerangka web (*framework*). File PHP dapat berisi text, HTML, CSS, Javascript, dan kode PHP. Kode PHP dieksekusi pada *server* dan hasilnya dikembalikan ke *browser* dalam bentuk HTML.

PHP dapat berjalan pada berbagai *platform* seperti Windows, Linux, Unix, Mac OS X, dan lainnya serta kompatibel dengan hampir seluruh *server* seperti Apache, IIS, dan lain sebagainya. PHP dapat menghasilkan konten web yang dinamis, membuat (*create*), membuka (*open*), menulis (*write*), menghapus (*delete*), dan menutup (*close*) file pada *server*. Selain itu, PHP dapat mengirim dan menerima *cookie*, menambah, menghapus, dan memodifikasi data pada *database*, mengontrol akses pengguna, dan juga dapat mengenkripsi data.

Saat seorang pengguna mengetik URL *http://example.org* pada sebuah *web client* (browser), browser tersebut mengirimkan sebuah GET *request* ke *server* (diasumsikan menggunakan Apache). Setelah menerima *request* tersebut, Apache kemudian mencari file bernama index.php dan memberikan file tersebut ke PHP interpreter untuk dijalankan. PHP membaca keseluruhan file dan mengeksekusi semua kode PHP yang ditemukan. Setelah selesai, PHP *intepreter* memberikan *output* dari kode kembali ke Apache. Apache kemudian mengirimkan *output* yang didapat dari PHP ke browser untuk ditampilkan pada layar pengguna.

Untuk membaca *header request*, PHP memiliki fungsi atau *method* getallheaders() untuk digunakan. Contoh kode untuk membaca *header request* adalah sebagai berikut.

```
foreach (getallheaders() as $name => $value)
{
    echo "$name: $valuen";
}
```

Contoh *output* dari kode tersebut adalah sebagai berikut.

```
Host: localhost
Connection: keep-alive
User-Agent:   Mozilla/5.0   (X11;   Linux   x86_64)
AppleWebKit/535.2      (KHTML,      like      Gecko)
Chrome/15.0.874.121 Safari/535.2
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.
9,*/*;q=0.8
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: UTF-8,*;q=0.5
Cookie:     username=admin;     password=21232ffc3;
PHPSESSID=o5m4e2td1m66c4pkjdag9vs0u2
```

Cara lain untuk membaca *header* adalah menggunakan http_get_*request*_headers() atau apache_*request*_headers(). Untuk membaca tipe atau *method* dari sebuah HTTP *request*, kita dapat menggunakan kode PHP $_*SERVER*[''*REQUEST*_METHOD]. Contoh penggunaannya adalah sebagai berikut.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
     // do something
 }
```

Untuk membaca *body request*, http_get_*request*_body() dapat digunakan jika *request method* adalah POST, selain itu menggunakan http_get_*request*_body_stream(). Contoh penggunaan adalah sebagai berikut.

```
 if ($body == NULL && $_SERVER['REQUEST_METHOD'] ==
'POST') {
        $body = http_get_request_body();
    }
```

Untuk mengirim *response header* pada PHP kita dapat menggunakan fungsi header(). Terdapat dua kasus spesial pemanggilan header. Yang pertama adalah berisi "HTTP/…" yang digunakan untuk mengirimkan HTTP *status code*, seperti contoh berikut.

```
<?php
   header("HTTP/1.0 404 Not Found");
?>
```

Yang kedua adalah *header* "Location:" yang mengirimkan *header* kembali ke browser dan mengembalikan sebuah *Redirect* dengan *status code* 302.

```
<?php
header("Location: http://www.example.com/"); /* Redi
rect browser */
```

```
/* Make sure that code below does not get executed w
hen we redirect. */
exit;
?>
```

Untuk membuat *response body* dapat menggunakan HttpResponse. Berikut adalah contoh penggunaan.

```
<?php
  HttpResponse::setCache(true);
  HttpResponse::setContentType('application/pdf');
  HttpResponse::setFile('sheet.pdf');
  HttpResponse::send();
?>
```

2.3. Protokol REST dan SOAP

Representational State Transfer (REST) dan *Simple Object Access Protocol* (SOAP) adalah arsitektur metode komunikasi pada *web service* yang umum digunakan pada protokol HTTP.

Pada protokol HTTP, terdapat 4 macam *request* yang dapat dibuat, sesuai dengan HTTP 1.1 verbs, yaitu [2]:

1. GET
2. POST
3. PUT
4. DELETE

REST Hanya dapat digunakan pada protokol HTTP dengan *request* berupa URL [5]. URL yang digunakan biasa disebut *endpoint* dalam pemanggilannya. Contoh penamaan URL/*endpoint* yang baik adalah: */user*, */user/1234*, */user/1234/photos*, */user/1234/photos/abc*.

Sedangkan Protokol SOAP menggunakan *eXtensible Markup Language* (XML) *structure* sebagai alat untuk membuat *request* [5]. Karena menggunakan XML, maka sebenarnya penggunaan protokol SOAP tidak hanya terbatas pada HTTP saja, akan tetapi dapat juga digunakan pada protokol-protokol lainnya yang mendukung XML.

Berikut ini adalah contoh sebuah SOAP *request* dalam bentuk XML [6]:

```
<?xml version = "1.0"?>
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV = "http://www.w3.org/2001/12/soap-
envelope"
  SOAP-ENV:encodingStyle =
"http://www.w3.org/2001/12/soap-encoding">

  <SOAP-ENV:Body xmlns:m =
"http://www.xyz.org/quotations">
    <m:GetQuotation>
      <m:QuotationsName>MiscroSoft</m:QuotationsName>
    </m:GetQuotation>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Pada protokol REST dan SOAP juga terdapat beberapa metode otentikasi. Metode otentikasi berguna untuk memastikan apakah *client* mempunyai hak untuk mengakses sebuah *endpoint*. Terdapat tiga macam metode otentikasi yang umum digunakan, yaitu Basic Authentication, Digest Authentication, dan OAuth [5].

Dalam melakukan transmisi data, *request* pada REST dan SOAP didahului oleh sebuah HTTP *header* yang berisi informasi-informasi yang diperlukan. Salah satu informasi pada *request header* yang digunakan adalah metode otentikasi.

Berikut ini adalah contoh *request* header dengan metode GET, menggunakan metode otentikasi Basic Authorization [7]:

```
GET / HTTP/1.1
Host: example.org
Authorization: Basic Zm9vOmJhcg==
```

2.4. *Scheduler*

Scheduler adalah sebuah perangkat lunak yang berfungsi untuk menjalankan suatu perintah/pekerjaan pada sebuah perangkat secara otomatis sesuai aturan yang ditetapkan. Dalam penggunaan *scheduler*, ada beberapa hal yang perlu ditetapkan

supaya *scheduler* dapat bekerja, yaitu: apa perintah yang akan dieksekusi, dan kapan perintah akan dijalankan [8].

Berikut ini adalah contoh implementasi *scheduler* sederhana pada bahasa pemrograman Java [9] .

```
import java.util.Timer;

public class SchedulerMain {
	public static void main(String args[]) throws
InterruptedException {

		Timer time = new Timer();
		ScheduledTask st = new ScheduledTask();
		time.schedule(st, 0, 1000);
		for (int i = 0; i < 3; i++) {
			System.out.println("Execution in
Main Thread..." + i);
			Thread.sleep(2000);
			if (i == 3) {
				System.exit(0);
			}
		}
	}
}
```

Potongan kode program di atas akan menghasilkan output sebagai berikut.

```
Execution in Main Thread....0
Time is :Tue Jun 19 14:21:42 IST 2012
Time is :Tue Jun 19 14:21:43 IST 2012
Execution in Main Thread....1
Time is :Tue Jun 19 14:21:44 IST 2012
Time is :Tue Jun 19 14:21:45 IST 2012
Execution in Main Thread....2
Time is :Tue Jun 19 14:21:46 IST 2012
Time is :Tue Jun 19 14:21:47 IST 2012
```

2.5. *Request* Response Time

Request Response Time adalah jeda waktu antara pengiriman *request* ke *web service* dan penerimaan response dari *web service*. Berikut ini adalah implementasi program untuk mengecek *Request Response Time* pada bahasa pemprograman PHP, dengan metode GET [10].

```
<?php
  if(!isset($_GET['url']))
  die("enter url");
  $ch=curl_init($_GET['url']); //get url
  curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
  if(curl_exec($ch))
  {
  $info = curl_getinfo($ch);
  echo
'Took '.$info['total_time'].' Seconds';
  }
  curl_close($ch);
?>
```

2.6. SQL Server

Microsoft SQL Server adalah sebuah sistem manajemen basis data relasional (RDBMS) produk Microsoft. Bahasa kueri utamanya adalah Transact-SQL yang merupakan implementasi dari SQL standar ANSI/ISO yang digunakan oleh Microsoft dan Sybase. Umumnya SQL Server digunakan di dunia bisnis yang memiliki basis data berskala kecil sampai dengan menengah, tetapi kemudian berkembang dengan digunakannya SQL Server pada basis data besar.

Microsoft SQL Server dan Sybase/ASE dapat berkomunikasi lewat jaringan dengan menggunakan protokol TDS (Tabular Data Stream). Selain itu, Microsoft SQL Server juga mendukung ODBC (Open Database Connectivity), dan mempunyai driver JDBC untuk

bahasa pemrograman Java. Fitur yang lain dari SQL Server ini adalah kemampuannya untuk membuat basis data *mirroring* dan *clustering*[11][12].

2.7. Kerangka Kerja CodeIgniter

CodeIgniter adalah sebuah kerangka kerja berbasis PHP yang dibangun dengan konsep MVC (Model, View, Controller) yang memiliki keunggulan yaitu memiliki *loading time* yang ringan, tersedianya sintaks yang ekspresif dan jelas sehingga dapat menghemat waktu, biaya awal, biaya pemeliharaan dan dapat meningkatkan kualitas perangkat lunak **Invalid source specified.Invalid source specified.**. Pada tugas akhir ini, PHP dengan kerangka kerja CodeIgniter akan digunakan untuk membangun aplikasi *Dashboard* pada Aplikasi Pemantauan *Web Service*. [13]

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi penjelasan mengenai analisis dan perancangan perangkat lunak untuk memenuhi fungsionalitas yang dibutuhkan dari tugas akhir. Perancangan ini meliputi perancangan data, perancangan proses, dan perancangan antar muka, serta analisis implementasi metode secara umum pada sistem.

3.1. Analisis Metode Secara Umum

Pada subbab berikut akan dijelaskan analisis perangkat lunak pada aplikasi pemantauan *web service*. Analisis yang dilakukan meliputi analisis permasalahan, kebutuhan umum perangkat lunak, deskripsi umum sistem, arsitektur dan kebutuhan fungsional sistem.

3.1.1. Analisis Permasalahan

Permasalahan utama yang diangkat pada pembuatan Tugas Akhir ini adalah bagaimana implementasi sistem perangkat lunak yang dapat melaksanakan pengujian pada *web service* dengan kasus uji secara berkala dengan menggunakan *scheduler*, serta menampilkan hasil pengujian endpoint pada *web service* secara berkala. Sistem ini dibutuhkan untuk mengawasi kinerja *endpoint* pada *web service* dari waktu ke waktu, untuk mempermudah *administrator* sistem.

Permasalahan ini dapat dibagi menjadi empat bagian yang lebih kecil, yaitu:

- Bagaimana rancangan arsitektur program yang efisien agar fungsi pengiriman *request* bisa terpenuhi,
- Bagaimana rancangan proses pengiriman kasus uji serta proses pada sistem *scheduler* yang sesuai untuk diaplikasikan pada permasalahan ini,

- Bagaimana rancangan basis data yang sesuai untuk diaplikasikan pada permasalahan ini,
- Dan bagaimana rancangan antarmuka yang mudah digunakan oleh pengguna.

3.1.2. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dibuat tiga buah perangkat lunak, yaitu berupa perangkat lunak pada *server* uji, perangkat lunak pada *server* pusat, dan perangkat lunak *dashboard* untuk melakukan pengaturan serta melihat hasil pengujian.

Perangkat lunak pada *server* uji berfungsi untuk melakukan pengujian pada *endpoint* target sesuai dengan kasus uji yang diberikan. Pengujian pada perangkat lunak ini diinisiasi oleh *request* yang dibuat oleh *server* pusat, menuju sebuah *endpoint* yang dimiliki oleh perangkat lunak pada *server* uji. Hasil pengujian yang diharapkan dari perangkat lunak ini adalah *response time* dan *response code* dari pengujian kasus uji.

Perangkat lunak pada *server* pusat berfungsi untuk menginisiasi pengujian pada *server* uji. Inisiasi pengujian dilakukan secara berkala oleh *scheduler* yang terdapat pada perangkat lunak ini.

Perangkat lunak *dashboard* berfungsi untuk melakukan pengaturan pada pengujian dan kasus uji. Selain itu, perangkat lunak *dashboard* dapat digunakan oleh pengguna untuk melihat hasil pengujian yang dilakukan.

3.1.3. Spesifikasi Kebutuhan Perangkat Lunak

Pada subbab ini, akan dibahas mengenai spesifikasi kebutuhan perangkat lunak dari hasil analisis yang telah dilakukan. Bagian ini berisi kebutuhan perangkat lunak yang direpresentasikan dalam bentuk kebutuhan fungsional, diagram kasus penggunaan dan diagram aktivitas.

3.1.3.1. Aktor

Pada perangkat lunak ini, terdapat empat aktor yang dapat masing-masing memiliki kasus penggunaan. Aktor-aktor tersebut yaitu Server Uji, Server Pusat, Pengguna dan Administrator.

3.1.3.2. Kebutuhan Fungsional

Kebutuhan fungsional mendefinisikan fungsi yang harus dimiliki oleh perangkat lunak, reaksi dari perangkat lunak terhadap suatu masukan, dan hasil yang diharapkan dari perangkat lunak. Terdapat tiga perangkat lunak yang terdapat pada sistem ini, yaitu perangkat lunak pada *server* uji, perangkat lunak pada *server* pusat dan aplikasi *dashboard.*

Kebutuhan fungsional dari perangkat lunak *server* uji dijelaskan pada Tabel 3.1.

Tabel 3.1 Daftar Kode Kebutuhan Fungsional Perangkat Lunak *Server* Uji

Kode Kebutuhan Fungsional	Kebutuhan Fungsional	Deskripsi
SU01	Mengirim *request* pada *endpoint* sesuai kasus uji	Mengirim *request* pada *endpoint* target sesuai dengan spesifikasi yang dibuat pada kasus uji.
SU02	Menyimpan data hasil pengujian	Memasukkan data hasil pengujian pada database

Kebutuhan fungsional dari perangkat lunak *server* pusat dijelaskan pada Tabel 3.2.

Tabel 3.2 Daftar Kode Kebutuhan Fungsional Perangkat Lunak *Server* Pusat

Kode Kebutuhan Fungsional	Kebutuhan Fungsional	Deskripsi
SP01	Mengirim perintah pengujian pada *Server* Uji	Mengirim perintah pengujian pada *Server* Uji secara berkala oleh *scheduler* pada *Server* Pusat

Kebutuhan fungsional dari perangkat lunak Aplikasi *Dashboard* secara umum dapat dimasukkan ke dalam dua kategori, yaitu kebutuhan fungisonal umum serta kebutuhan fungsional *administrator*. Kebutuhan fungsional umum memiliki kode "DU", dan kebutuhan fungsional admin memiliki kode "DA". Kebutuhan fungsional umum juga dimiliki oleh *administrator*, namun tidak sebaliknya. Kebutuhan fungsional dari perangkat lunak *Dashboard* dijelaskan pada Tabel 3.3.

Tabel 3.3 Daftar Kode Kebutuhan Fungsional Perangkat Lunak Aplikasi *Dashboard*

Kode Kebutuhan Fungsional	Kebutuhan Fungsional	Deskripsi
DU01	Menambah kasus uji	Menambah data kasus uji dan mengatur spesifikasi kasus uji
DU02	Menampilkan daftar kasus uji	Menampilkan daftar kasus uji milik pengguna, dan daftar kasus uji milik pengguna lain yang bersifat publik
DU03	Melakukan perubahan pada kasus uji	Melakukan perubahan pada detail kasus uji yang telah dipasang

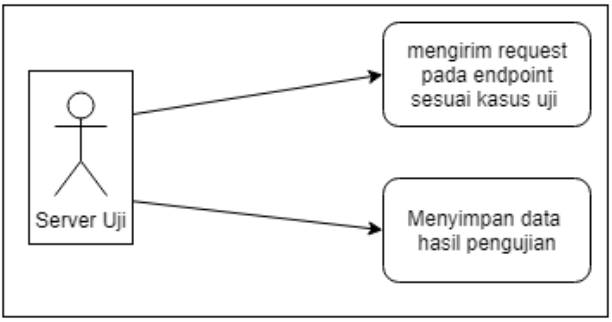
DU04	Nonaktifkan kasus uji	Menonaktifkan kasus uji sehingga di kemudian hari tidak dapat dipasang kembali. Pemasangan yang sudah ada tidak hilang.
DU05	Menampilkan daftar pengujian pada *Server* Uji	Menampilkan daftar pengujian pada *Server* Uji tertentu
DU06	Menambahkan pengujian dengan kasus uji pada *Server* Uji	Menambahkan pengujian dengan kasus uji pada *Server* Uji dan mengatur spesifikasi pengujian
DU07	Mengubah pengaturan pengujian pada *Server* Uji	Merubah pengaturan pengujian pada *Server* Uji

DU08	Menampilkan hasil pengujian pada *Server* Uji	Menampilkan tabel hasil pengujian dan grafik hasil pengujian secara *real-time*
DU09	Nonaktifkan pengujian pada *Server* Uji	Menonaktifkan pengujian pada *Server* Uji. Pengujian tetap ada, namun tidak lagi dieksekusi oleh *Server* Uji.
DA01	Memberi akses pengguna ke *Server* Uji	Administrator menentukan *Server* Uji mana yang dapat diakses oleh pengguna

3.1.3.3. Diagram Kasus Penggunaan

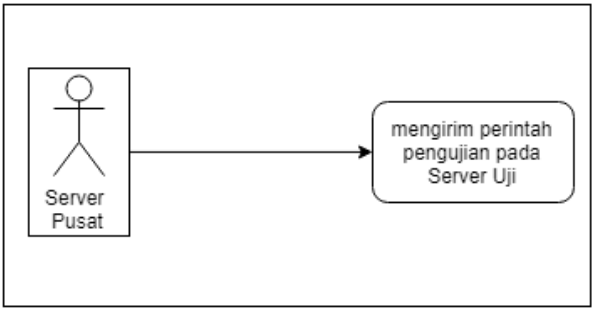
Pada subbab ini akan dibahas kasus penggunaan perangkat lunak secara rinci. Daftar kasus penggunaan ditentukan berdasarkan pada hasil analisis kebutuhan fungsional dari perangkat lunak. Terdapat dua kasus penggunaan pada perangkat lunak *Server* Uji, satu kasus penggunaan pada perangkat lunak *Server* Pusat, dan sepuluh kasus penggunaan pada perangkat lunak *Dashboard*.

Daftar kasus penggunaan pada perangkat lunak *Server* Uji terdapat pada gambar 3.1.



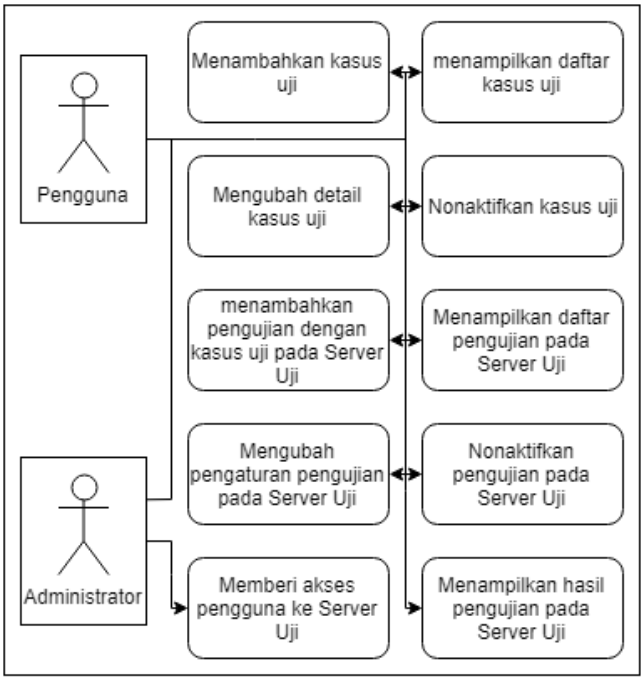
Gambar 3.1 Kasus Penggunaan pada Perangkat Lunak *Server* Uji

Daftar kasus penggunaan pada perangkat lunak *Server* Pusat terdapat pada gambar 3.2.



Gambar 3.2 Kasus Penggunaan pada Perangkat Lunak *Server* Pusat

Daftar kasus penggunaan pada perangkat lunak *Server* Uji terdapat pada gambar 3.3.



Gambar 3.3 Kasus Penggunaan pada Perangkat Lunak *Dashboard*

Keterangan lebih lanjut mengenai kode dan deskripsi kasus penggunaan dapat dilihat pada tabel 3.4.

Tabel 3.4 Deskripsi Kasus Penggunaan Sistem

Kode Kasus Penggunaan	Nama Kasus Penggunaan
SU01	Mengirim *request* pada *endpoint* sesuai kasus uji
SU02	Menyimpan data hasil pengujian ke database

SP01	Mengirim perintah pengujian pada *Server* Uji
DU01	Menambah kasus uji
DU02	Menampilkan daftar kasus uji
DU03	Mengubah detail kasus uji
DU04	Nonaktifkan kasus uji
DU05	Menampilkan daftar pengujian pada *Server*
DU06	Menambahkan pengujian dengan kasus uji
DU07	Mengubah pengaturan pengujian pada *Server* Uji
DU08	Menampilkan hasil pengujian dari pemasangan kasus uji pada *Server* Uji
DU09	Nonaktifkan pengujian pada *Server* Uji
DA01	Memberi akses pengguna ke *Server* Uji

3.1.3.4. Pembahasan Kasus Penggunaan

Kasus penggunaan pada sub bab sebelumnya akan dibahas pada sub bab ini. Akan dibahas mengenai dua hal untuk setiap pembahasan, yaitu rincian alur kasus penggunaan dan diagram aktivitas kasus penggunaan.

3.1.3.4.1. Kasus Penggunaan Mengirim *Request* pada *Endpoint* Sesuai Kasus Uji

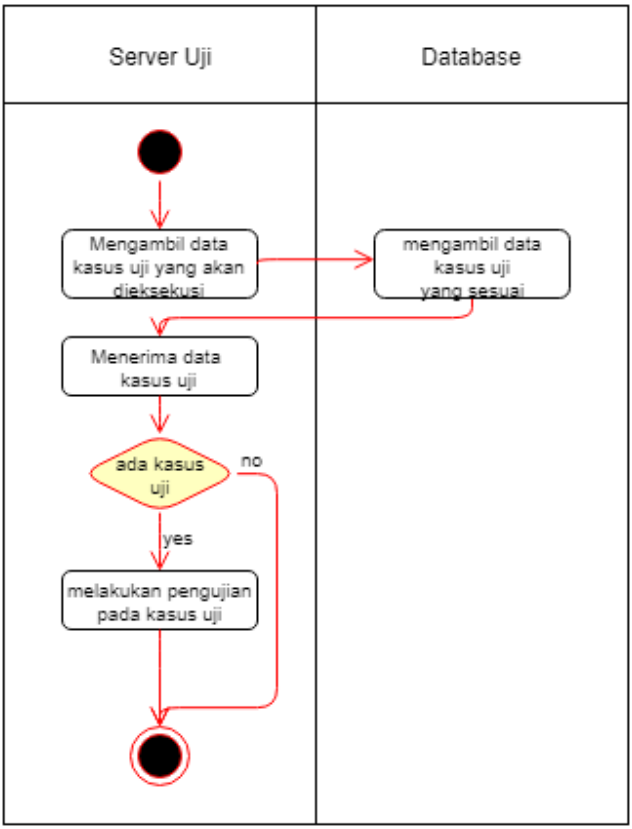
Kasus Penggunaan ini terdapat pada aktor *Server* Uji. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.5.

Tabel 3.5 Rincian Alur Kasus Penggunaan Mengirim *Request* pada *Endpoint* Sesuai Kasus Uji

Nama Kasus Penggunaan	Mengirim *Request* pada *Endpoint* Sesuai Kasus Uji
Nomor	SU01
Aktor	*Server* Uji
Kondisi Awal	Perintah pengiriman *request* telah diterima dari *Server* Pusat
Kondisi Akhir	*Request* dieksekusi oleh *Server* Uji
Alur Normal	1. *Server* Uji mencari data pengujian mana saja yang sesuai dengan parameter perintah pengujian 2. *Server* Uji mengambil data pengujian yang sesuai dari database 3. *Server* Uji mengeksekusi *request* sesuai data-data yang diambil dari database.
Alur Alternatif	A1. Tidak ada data pengujian yang sesuai. A1.1. *Server* Uji tidak mengirim *request* apapun.

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.5, Alur tersebut dapat dikonversi menjadi diagram aktivitas.

Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Aktivitas Kasus Penggunaan Mengirim *Request* pada *Endpoint* sesuai Kasus Uji

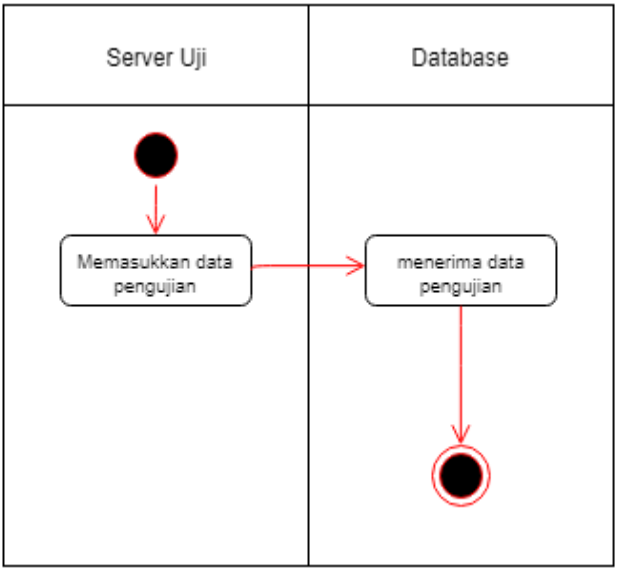
3.1.3.4.2. Kasus Penggunaan Menyimpan Data Hasil Pengujian ke Database

Kasus Penggunaan ini terdapat pada aktor *Server* Uji. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.6.

Tabel 3.6 Rincian Alur Kasus Penggunaan Menyimpan Data Hasil Pengujian ke Database

Nama Kasus Penggunaan	Menyimpan data hasil pengujian ke database
Nomor	SU02
Aktor	*Server* Uji
Kondisi Awal	Eksekusi pengiriman sebuah kasus uji telah selesai dieksekusi
Kondisi Akhir	Data hasil pengujian tersimpan di database
Alur Normal	1. *Server* Uji mengirim data hasil pengujian ke database 2. Database menyimpan data hasil pengujian
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.6, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.5.



Gambar 3.5 Diagram Aktivitas Kasus Penggunaan Menyimpan data hasil pengujian ke database

3.1.3.4.3. Kasus Penggunaan Mengirim Perintah Pengujian pada *Server* Uji

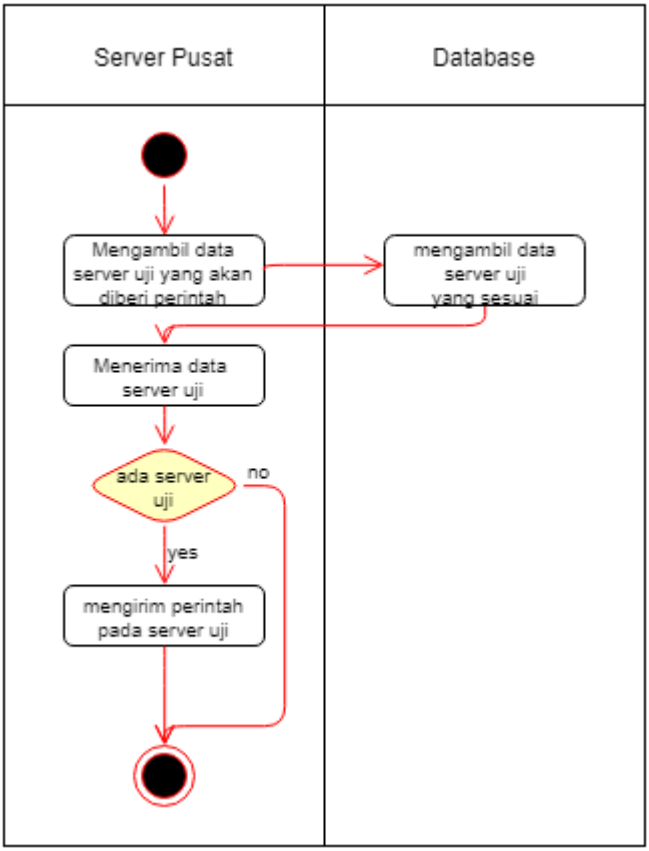
Kasus Penggunaan ini terdapat pada aktor *Server* Pusat. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.7.

Tabel 3.7 Rincian Alur Kasus Penggunaan Mengirim perintah pengujian pada *Server* Uji

Nama Kasus Penggunaan	Mengirim perintah pengujian pada *Server* Uji
Nomor	SP01
Aktor	*Server* Pusat

Kondisi Awal	*Scheduler* pada *Server* Pusat menginisiasi pengiriman perintah pengujian
Kondisi Akhir	*Server* Uji mengeksekusi pengujian
Alur Normal	1. *Server* Pusat mencari data *Server* Uji yang akan diperintah pada database 2. Database memberikan daftar *server* uji yang akan diberi diperintah 3. *Server* Pusat mengirim perintah pengujian ke setiap *Server* Uji yang ada pada daftar
Alur Alternatif	A1. Tidak ada *server* uji yang harus diberi perintah A1.1. *Server* Pusat tidak mengirim perintah apapun

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.7, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.6.



Gambar 3.6 Diagram Aktivitas Kasus Penggunan Mengirim perintah pengujian pada *Server* Uji

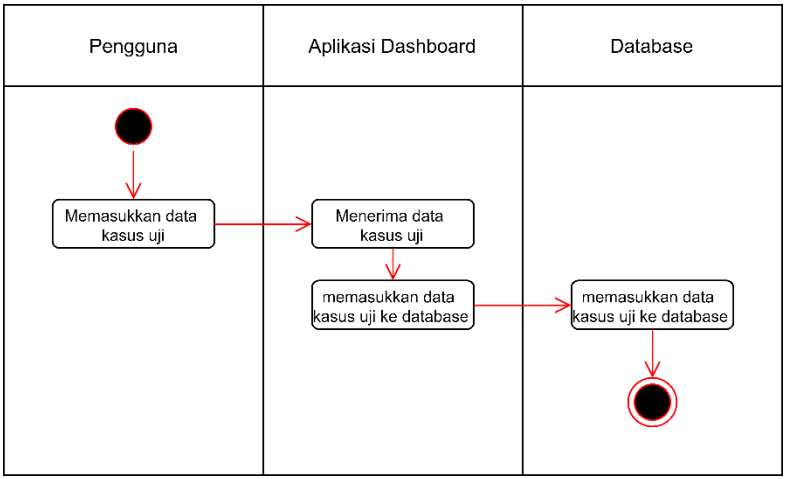
3.1.3.4.4. Kasus Penggunaan Menambah Kasus Uji

Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.8.

Tabel 3.8 Rincian Alur Kasus Penggunaan Menambah Kasus Uji

Nama Kasus Penggunaan	Menambah Kasus Uji
Nomor	DU01
Aktor	Pengguna
Kondisi Awal	Kasus Uji belum ditambahkan
Kondisi Akhir	Kasus Uji telah ditambahkan
Alur Normal	1. Pengguna Memasukkan data Kasus Uji 2. Aplikasi *Dashboard* menerima data kasus uji 3. Aplikasi *Dashboard* memasukkan data kasus uji ke database 4. Data kasus uji tersimpan di database
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.8, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 Diagram Aktivitas Kasus Penggunaan Menambah Kasus Uji

3.1.3.4.5. Kasus Penggunaan Menampilkan Daftar Kasus Uji

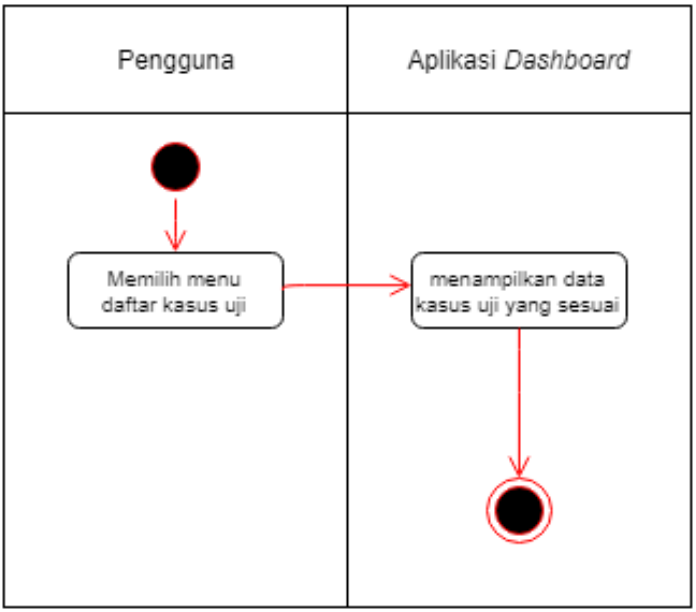
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.9.

Tabel 3.9 Rincian Alur Kasus Penggunaan Menampilkan daftar Kasus Uji

Nama Kasus Penggunaan	Menampilkan daftar Kasus Uji
Nomor	DU02
Aktor	Pengguna
Kondisi Awal	Pengguna akan melihat daftar kasus uji

Kondisi Akhir	Aplikasi *Dashboard* menampilkan daftar kasus uji
Alur Normal	1. Pengguna membuka menu untuk melihat daftar kasus uji 2. Sistem menampikan daftar kasus uji yang dapat dilihat oleh pengguna tersebut
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.9, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram Aktivitas Kasus Penggunaan Menampilkan daftar Kasus Uji

3.1.3.4.6. Kasus Penggunaan Mengubah Detail Kasus Uji

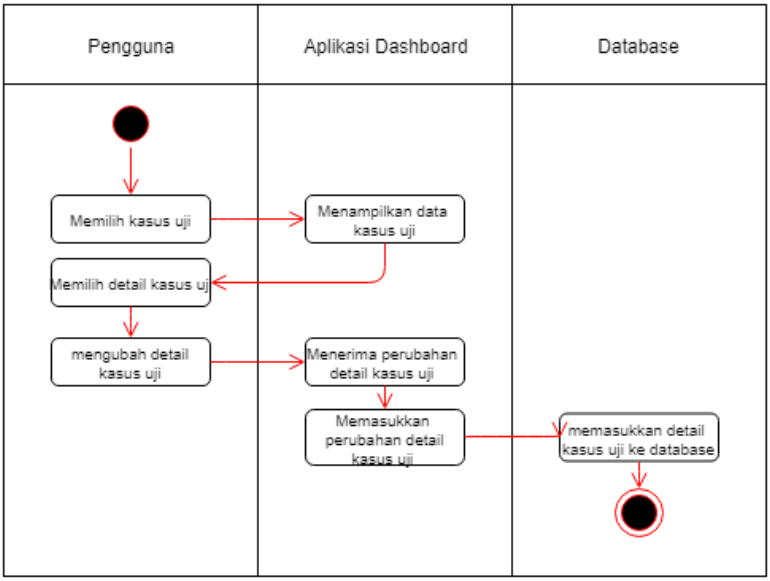
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.10.

Tabel 3.10 Rincian Alur Kasus Penggunaan Mengubah Detail Kasus Uji

Nama Kasus Penggunaan	Kasus Penggunaan Mengubah Detail Kasus Uji
Nomor	DU03

Aktor	Pengguna
Kondisi Awal	Pengguna akan merubah detail kasus uji
Kondisi Akhir	Perubahan detail kasus uji disimpan pada *database*
Alur Normal	1. Pengguna memilih kasus uji 2. Aplikasi *Dashboard* menampilkan detail kasus uji 3. Pengguna memilih detail kasus uji yang akan diubah 4. Pengguna mengubah detail kasus uji 5. Aplikasi *Dashboard* menerima perubahan detail kasus uji 6. Aplikasi *Dashboard* memasukkan perubahan detail kasus uji ke database 7. Perubahan detail kasus uji tersimpan di database
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.10, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.9.



Gambar 3.9 Diagram Aktivitas Kasus Penggunaan Mengubah Detail Kasus Uji

3.1.3.4.7. Kasus Penggunaan Menonaktifkan Kasus Uji

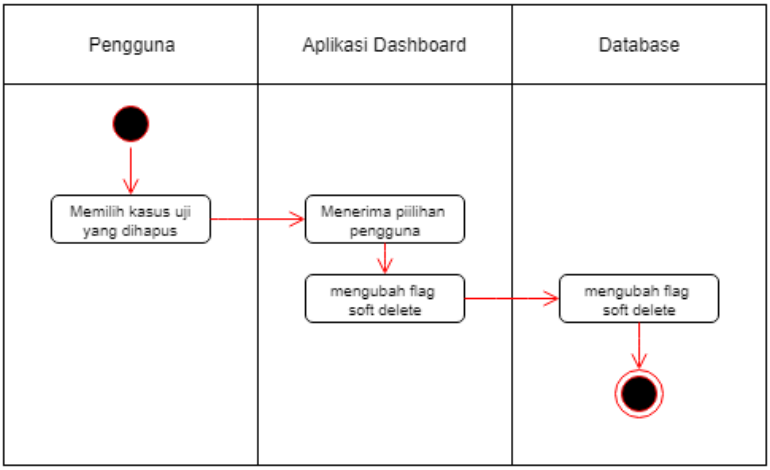
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.11.

Tabel 3.11 Rincian Alur Kasus Penggunaan Nonaktifkan Kasus Uji

Nama Kasus Penggunaan	Kasus Penggunaan Nonaktifkan Kasus Uji
Nomor	DU04
Aktor	Pengguna

Kondisi Awal	Pengguna akan menonaktifkan kasus uji
Kondisi Akhir	Status kasus uji berubah menjadi nonaktif
Alur Normal	1. Pengguna memilih kasus uji yang akan dihapus 2. Aplikasi Dashbord menerima pilihan pengguna 3. Aplikasi *Dashboard* mengganti status kasus uji menjadi nonaktif 4. Status kasus uji pada database menjadi nonaktif
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.11, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.10.



Gambar 3.10 Diagram Aktivitas Kasus Penggunaan Nonaktifkan Kasus Uji

3.1.3.4.8. Kasus Penggunaan Menampilkan Daftar Pengujian pada *Server* Uji

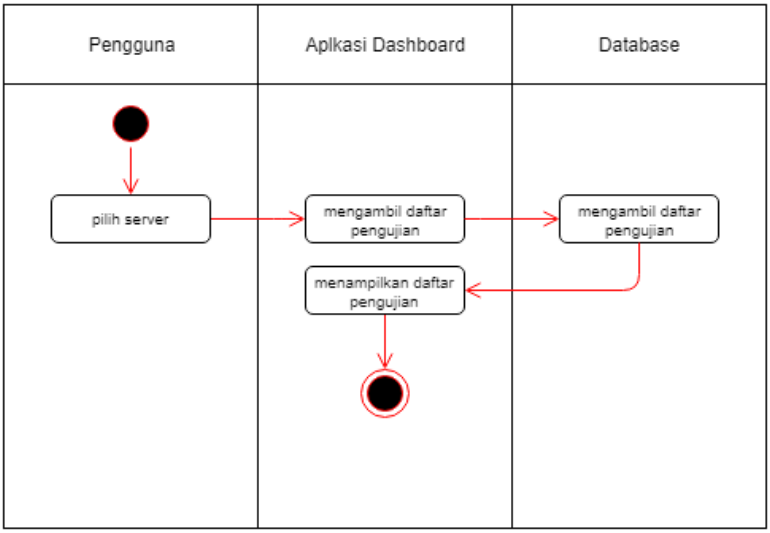
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.12.

Tabel 3.12 Rincian Alur Kasus Penggunaan Menampilkan daftar pengujian pada *Server* Uji

Nama Kasus Penggunaan	Kasus Penggunaan Menampilkan daftar pengujian pada *Server* Uji
Nomor	DU05
Aktor	Pengguna
Kondisi Awal	Pengguna akan Menampilkan daftar pengujian pada *Server* Uji

Kondisi Akhir	Sistem akan menampilkan daftar pengujian pada *Server* Uji yang dipilih
Alur Normal	1. Pengguna memilih *Server* Uji 2. Aplikasi mengambil dan menampilkan daftar pengujian pada *Server* Uji
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.12, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.11.



Gambar 3.11 Diagram Aktivitas Kasus Penggunaan Menampilkan Daftar Pengujian pada *Server* Uji

3.1.3.4.9. Kasus Penggunaan Menambahkan Pengujian dengan Kasus Uji pada *Server* Uji

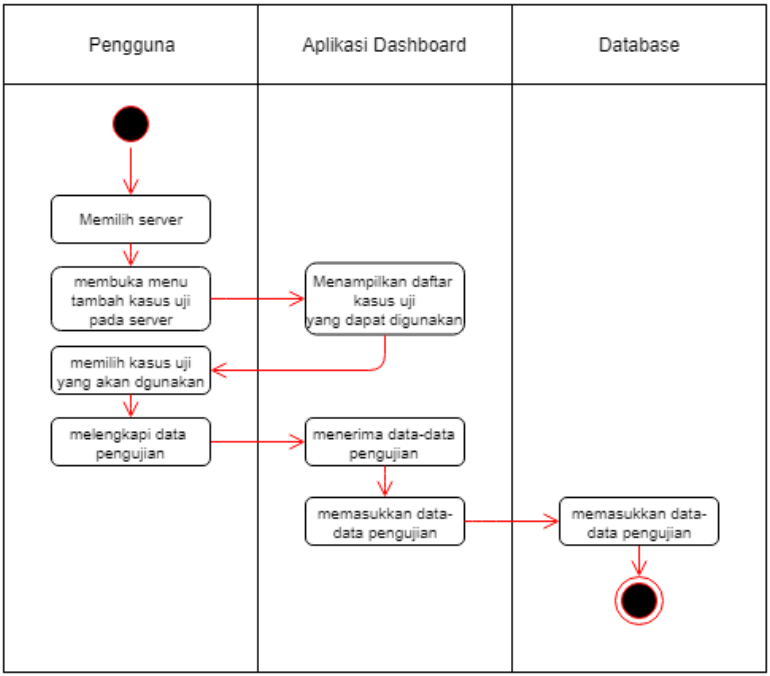
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.13.

Tabel 3.13 Rincian Alur Kasus Penggunaan Menambahkan Pengujian dengan Kasus Uji pada *Server* Uji

Nama Kasus Penggunaan	Menambahkan pengujian dengan kasus uji pada *Server* Uji
Nomor	DU06
Aktor	Pengguna

Kondisi Awal	Pengguna akan menambahkan pengujian ke *Server* Uji dengan kasus uji yang sudah ada.
Kondisi Akhir	Pengujian baru telah dipasang pada *Server* Uji
Alur Normal	1. Pengguna memilih *Server* Uji 2. Pengguna memilih menu tambah pengujian pada *Server* Uji 3. Daftar kasus uji yang bisa digunakan akan ditampilkan 4. Pengguna melengkapi data pengaturan pengujian 5. Aplikasi menerima data pengaturan pengujian dari pengguna 6. Aplikasi memasukkan data pengujian yang baru ke database 7. Data pengujian yang baru tersimpan di database
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.13, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.12.



Gambar 3.12 Diagram Aktivitas Kasus Penggunaan Menambahkan Pengujian dengan Kasus Uji pada *Server* Uji

3.1.3.4.10. Kasus Penggunaan Mengubah Pengaturan Pengujian pada *Server* Uji

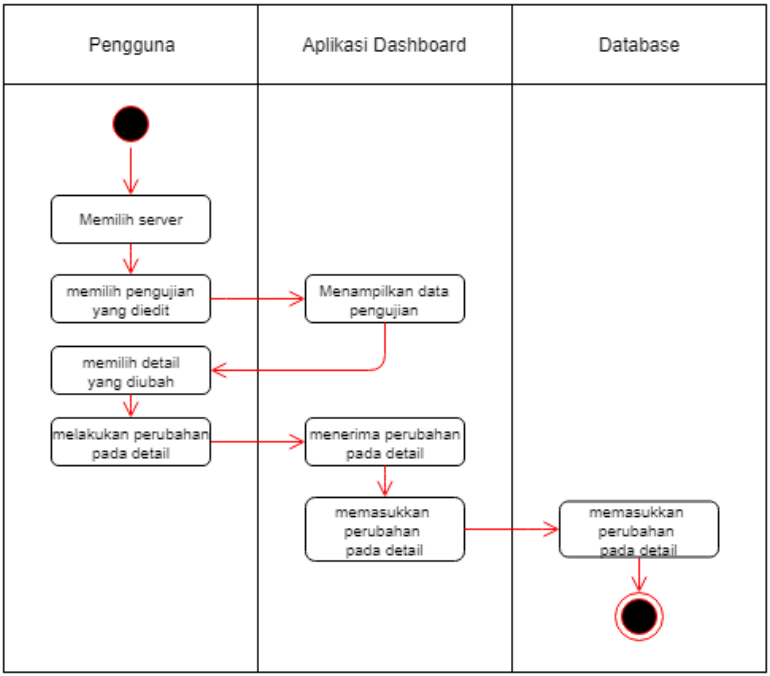
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.14.

Tabel 3.14 Rincian Alur Kasus Penggunaan Mengubah Pengaturan Pengujian pada *Server* Uji

Nama Kasus Penggunaan	Kasus Penggunaan Mengubah pengaturan pengujian pada *Server* Uji

Nomor	DU07
Aktor	Pengguna
Kondisi Awal	Pengguna akan merubah detail pengujian
Kondisi Akhir	Perubahan detail pengujian tersimpan pada database
Alur Normal	1. Pengguna memilih *Server* Uji 2. Pengguna memilih pengujian 3. Aplikasi Dashbord menampilkan detail pengujian 4. Pengguna memilih detail pengujian yang akan diubah 5. Pengguna mengubah detail pengujian 6. Aplikasi menerima perubahan detail pengujian 7. Aplikasi memasukkan perubahan detail ke database 8. Perubahan detail pengujian tersimpan pada database
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.14, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.13.



Gambar 3.13 Diagram Aktivitas Mengubah Pengaturan Pengujian pada *Server* Uji

3.1.3.4.11. Kasus Penggunaan Menampilkan Hasil Pengujian pada *Server* Uji

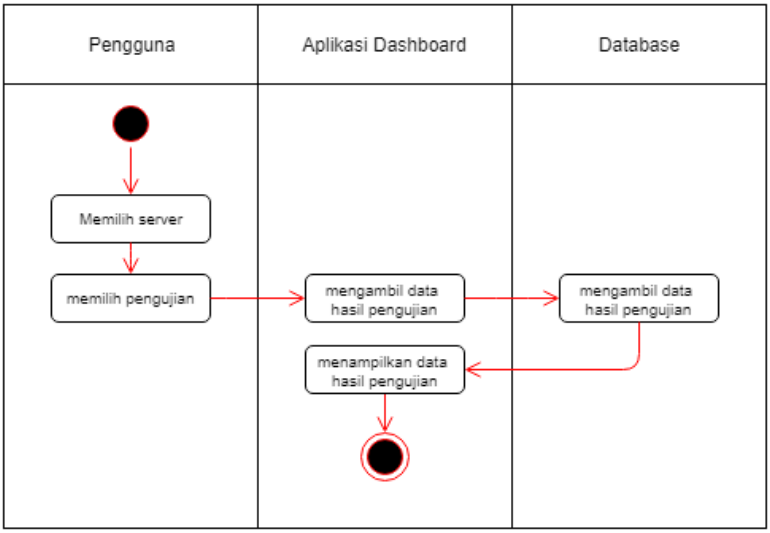
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.15.

Tabel 3.15 Rincian Alur Kasus Penggunaan Menampilkan Hasil Pengujian pada *Server* Uji

Nama Kasus Penggunaan	Kasus Penggunaan Menampilkan hasil pengujian pada *Server* Uji

Nomor	DU08
Aktor	Pengguna
Kondisi Awal	Pengguna akan melihat hasil pengujian pada *Server* Uji
Kondisi Akhir	Sistem menampilkan hasil pengujian dalam bentuk grafik dan tabel
Alur Normal	1. Pengguna memilih *Server* Uji 2. Pengguna memilih pengujian yang akan ditampilkan 3. Aplikasi akan menampilkan data hasil pengujian
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.15, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.14.



Gambar 3.14 Diagram Aktivitas Kasus Penggunaan Menampilkan Hasil Pengujian pada *Server* Uji

3.1.3.4.12. Kasus Penggunaan Nonaktifkan Pengujian

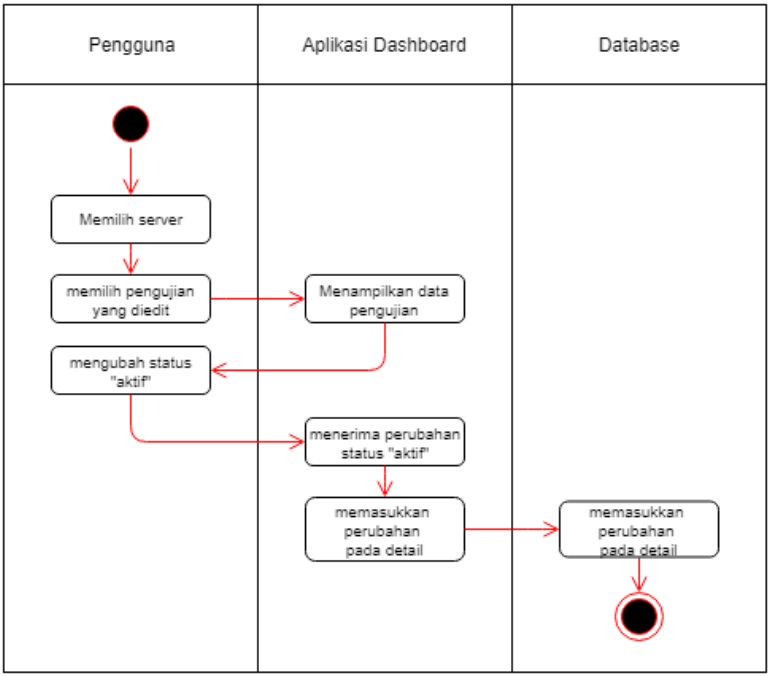
Kasus Penggunaan ini terdapat pada aktor Pengguna. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.16

Tabel 3.16 Rincian Alur Kasus Penggunaan Nonaktifkan Pengujian

Nama Kasus Penggunaan	Kasus Penggunaan Nonaktifkan Pengujian
Nomor	DU09
Aktor	Pengguna

Kondisi Awal	Pengguna akan menonaktifkan pengujian
Kondisi Akhir	Status pengujian berubah menjadi nonaktif
Alur Normal	1. Pengguna memilih *Server* Uji 2. Pengguna memilih pengujian 3. Aplikasi Dashbord menampilkan detail pengujian 4. Pengguna mengubah detail pengujian “aktif” 5. Aplikasi menerima perubahan detail pengujian 6. Aplikasi memasukkan perubahan detail ke database 7. Perubahan detail pengujian tersimpan pada database
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.16, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.15.



Gambar 3.15 Diagram Aktivitas Kasus Penggunaan Nonaktifkan Pengujian

3.1.3.4.13. Kasus Penggunaan Memberi Akses Pengguna ke *Server* Uji

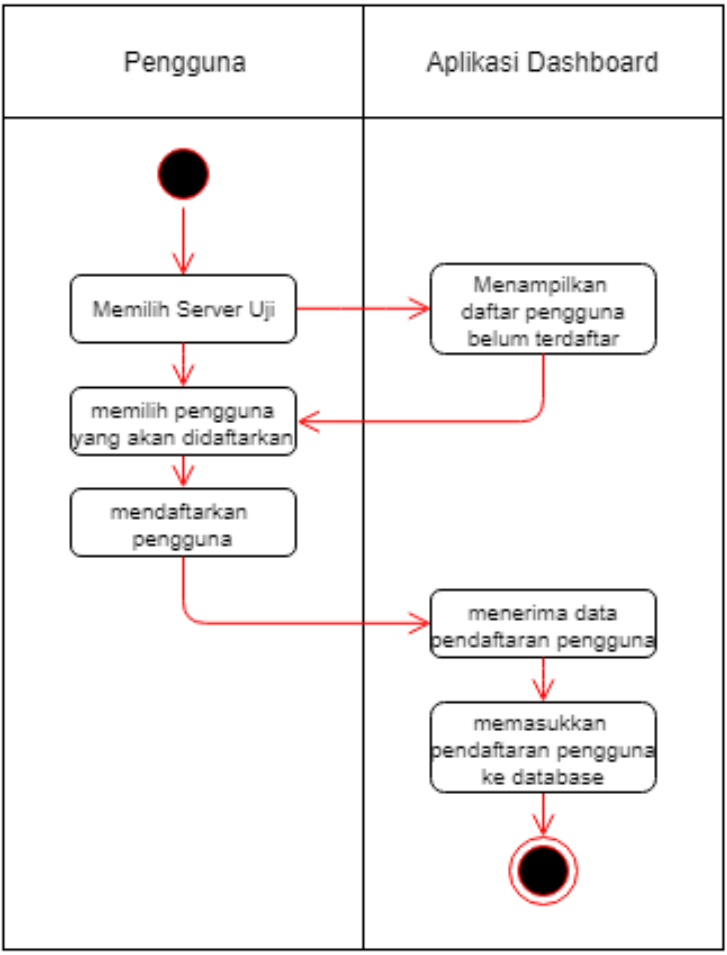
Kasus Penggunaan ini terdapat pada aktor Administrator. Penjelasan lebih rinci mengenai alur kasus uji ini dapat dilihat pada tabel 3.17

Tabel 3.17 Rincian Alur Kasus Penggunaan Memberi Akses Pengguna ke *Server* Uji

Nama Kasus Penggunaan	Kasus Penggunaan Memberi Akses Pengguna ke *Server* Uji

Nomor	DA01
Aktor	Administrator
Kondisi Awal	Administrator akan mendaftarkan akses pengguna ke *server* uji
Kondisi Akhir	Pengguna terdaftar pada *Server* Uji
Alur Normal	1. Administrator memilih *Server* Uji 2. Sistem menampilkan data pengguna yang belum terdaftar 3. Administrator memilih pengguna yang akan didaftarkan 4. Administrator mendaftarkan pengguna pada *Server* Uji 5. Sistem menyimpan pendaftaran pengguna pada *server* ini
Alur Alternatif	-

Berdasarkan deskripsi alur kasus penggunaan pada Tabel 3.17, Alur tersebut dapat dikonversi menjadi diagram aktivitas. Diagram aktivitas untuk kasus penggunaan ini dapat dilihat pada Gambar 3.16.



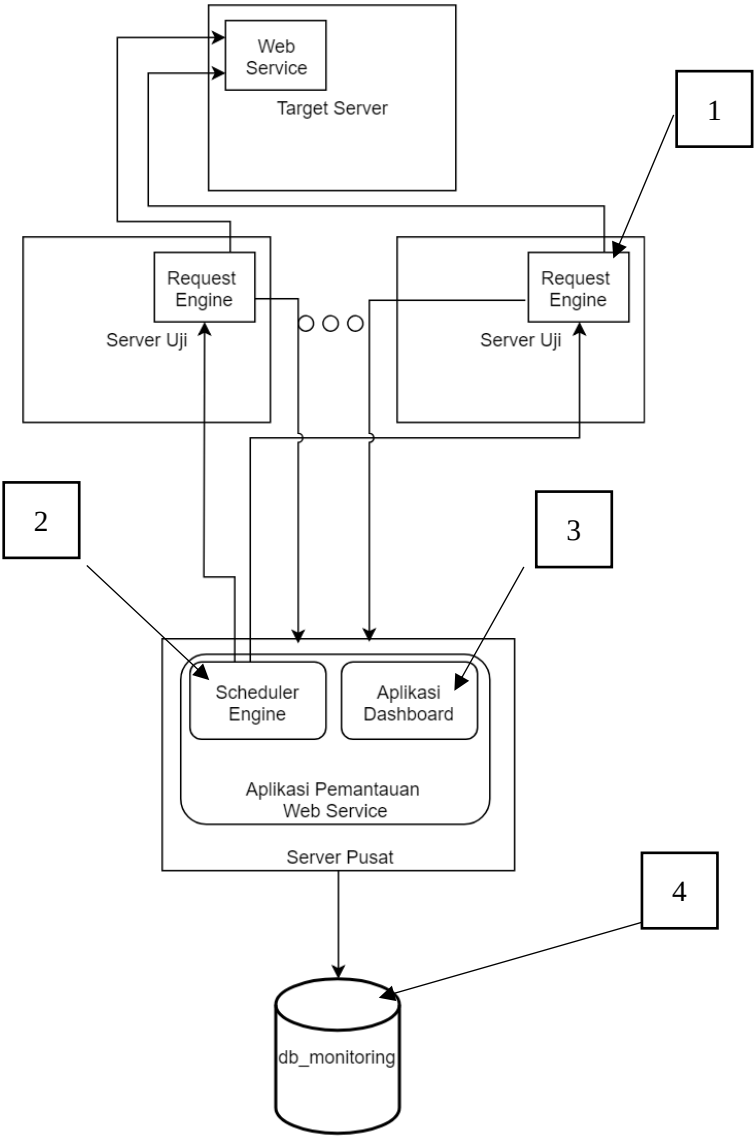
Gambar 3.16 Diagram Aktivitas Kasus Penggunaan Memberi Akses Pengguna ke *Server* Uji

3.2. Perancangan Sistem

Pada subbab ini dijelaskan mengenai tahapan perancangan sistem. Perancangan sistem ini dibagi menjadi beberapa bagian yang meliputi perancangan arsitektur sistem, perancangan proses aplikasi, dan perancangan antarmuka pengguna, dan perancangan basis data.

3.2.1. Perancangan Arsitektur

Dalam tugas akhir ini akan dirancang dan diaplikasikan sebuah sistem dengan banyak perangkat keras yang dapat dikategorikan menjadi dua kategori, yaitu *Server* Uji dan *Server* Pusat. Secara garis besar aplikasi ini memiliki rancangan arsitektur sistem yang dapat dilihat pada Gambar 3.17.



Gambar 3.17 Diagram Arsitektur Aplikasi Pemantauan *Web Service*

Pada gambar 3.17 yang ditunjuk oleh panah nomor 1, terdapat bagian perangkat lunak pada *Server Uji*. Perangkat lunak ini berfungsi untuk mengeksekusi pengujian dengan kasus uji yang ada. Pada sistem ini, dimungkinkan adanya lebih dari satu *Server Uji*. Hal ini memungkinkan sistem untuk mengeksekusi pengujian dengan beberapa *Server* Uji sekaligus. Perangkat lunak ini mengeksekusi pengujian hanya ketika mendapat perintah eksekusi pengujian dari perangkat lunak pada *Server* Pusat.

Pada gambar 3.17 yang ditunjuk oleh panah nomor 2, terdapat bagian perangkat lunak pada *Server* Pusat. Perangkat lunak ini berfungsi untuk mengirimkan perintah eksekusi pengujian kepada *Server* Uji yang terdaftar. Perintah eksekusi pengujian ini dikirimkan secara berkala setiap menit oleh *Server* Pusat kepada *Server* Uji. Fungsi pengiriman secara berkala oleh *Server* Pusat ditangani oleh fungsi *scheduler* yang terdapat pada perangkat lunak *Server* Pusat.

Pada gambar 3.17 yang ditunjuk oleh panah nomor 3, terdapat bagian perangkat lunak aplikasi *Dashboard*. Perangkat lunak ini berfungsi sebagai tampilan antarmuka yang dapat diakses oleh pengguna sistem. Pada perangkat lunak ini, pengguna dapat membuat pengaturan-pengaturan pada kasus uji dan pengujian yang dieksekusi oleh *Server* Uji dan *Server* Pusat.

Pada gambar 3.17 yang ditunjuk oleh panah nomor 4, terdapat bagian basis data. Basis data ini berfungsi untuk menyimpan data-data yang ada pada sistem.

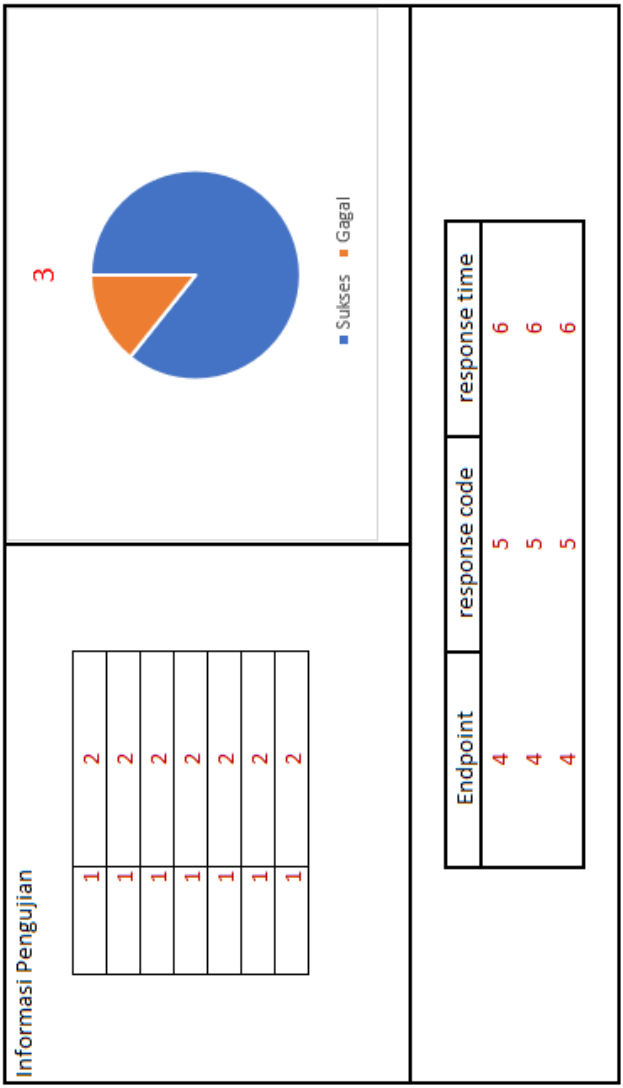
3.2.2. Perancangan Antarmuka

Dari tiga perangkat lunak yang terdapat di sistem ini, hanya satu yang memiliki bagian antarmuka yang berhubungan langsung dengan pengguna, yaitu aplikasi *Dashboard*.

Pada bagian ini dijelaskan mengenai rancangan tampilan antarmuka pengguna pada sistem. Pada subbab-subbab berikut akan dijelaskan masing-masing rancangan antarmuka yang terdapat pada sistem.

3.2.2.1. Antarmuka Halaman Pembuka

Antarmuka halaman pembuka (*Dashboard*) merupakan antarmuka yang pertama kali ditemui oleh pengguna. Halaman ini berisi catatan-catatan mengenai data-data pengujian yang ada. Rancangan antarmuka halaman ini dapat dilihat pada Gambar 3.18.



Gambar 3.18 Rancangan Antarmuka Halaman Pembuka

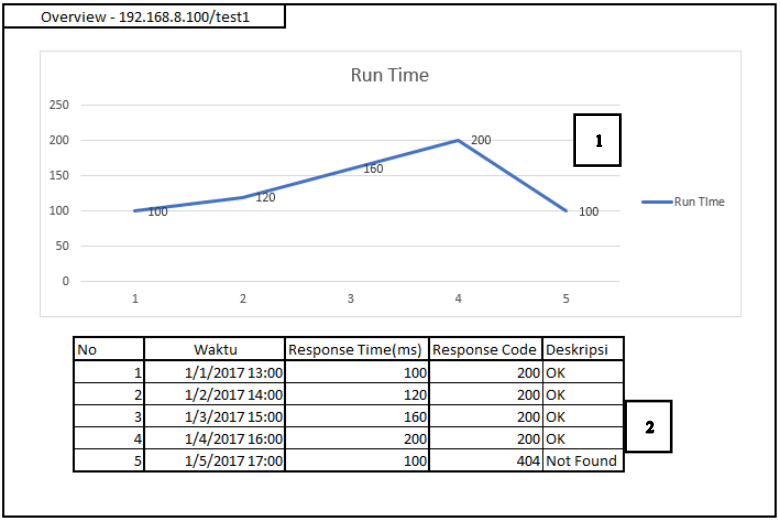
Elemen yang memuat bagian nomor 1, 2 dan 3 adalah bagian yang berfungsi untuk menampilkan informasi-informasi pengujian, sementara elemen yang memuat bagian nomor 4, 5 dan 6 merupakan tabel yang menampilkan beberapa pengujian terakhir pada sistem. Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada tabel 3.18.

Tabel 3.18 Bagian-Bagian Antarmuka Halaman Pembuka (*Dashboard*)

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Jenis informasi pengujian	*text*	Menampilkan jenis informasi pada pengujian.	-
2	Data informasi pengujian	*text*	Menampilkan data informasi pengujian sesuai jenis pada nomor 1.	-
3	Grafik jumlah pengujian berhasil dan gagal	*Pie chart*	Merepresentasikan Hasil pengujian gagal dan berhasil dalam bentuk *pie chart*	-
4	*Endpoint* Pengujian	*text*	Kolom *endpoint* pengujian pada tabel	-
5	*Response code*	*text*	Kolom hasil *response code* pada tabel	-
6	*Response time*	*text*	Kolom hasil *response time* pada tabel	-

3.2.2.2. Antarmuka Halaman Laporan Hasil Pengujian

Antarmuka halaman laporan hasil pengujian merupakan antarmuka yang berfungsi untuk menampilkan grafik hasil pengujian yang telah dipasang. Rancangan antarmuka halaman laporan hasil pengujian dapat dilihat pada Gambar 3.19.



No	Waktu	Response Time(ms)	Response Code	Deskripsi
1	1/1/2017 13:00	100	200	OK
2	1/2/2017 14:00	120	200	OK
3	1/3/2017 15:00	160	200	OK
4	1/4/2017 16:00	200	200	OK
5	1/5/2017 17:00	100	404	Not Found

Gambar 3.19 Rancangan Halaman Antarmuka Laporan Hasil Pengujian

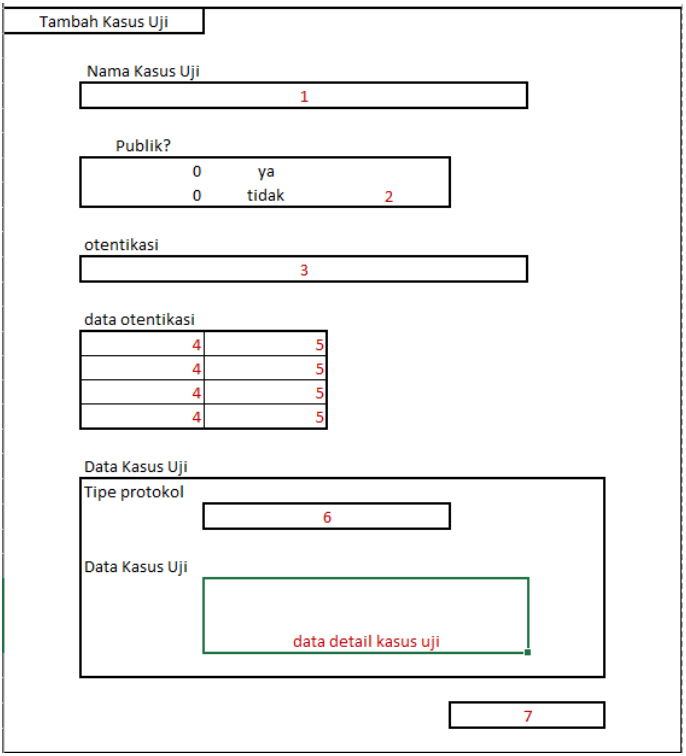
Bagian nomor 1 pada gambar 3.19 adalah bagian antarmuka yang berguna untuk menampilkan grafik *response time* pada pengujian kasus uji dari waktu ke waktu. Jika ada pengujian yang baru dieksekusi, sistem akan menambahkan data secara otomatis pada grafik ini.

Bagian nomor 2 pada gambar 3.19 adalah bagian antarmuka yang berfungsi untuk menampilkan data pengujian dalam bentuk tabel. Jika ada pengujian yang baru dieksekusi, sistem akan menambahkan data secara otomatis pada tabel ini.

Pada tabel yang ditandai dengan nomor 2, data yang ditampilkan yaitu waktu pengujian, *response time*, *response code* dan deskripsi untuk *response code* yang sesuai.s

3.2.2.3. Antarmuka Halaman Penambahan Kasus Uji

Antarmuka halaman penambahan kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk memasukkan kasus uji pada sistem. Terdapat 3 bagian rancangan antarmuka pada halaman ini, yaitu bagian rancangan halaman keseluruhan, penambahan data kasus uji REST *request*, dan penambahan data kasus uji SOAP *request*. Masing-masing rancangan antarmuka halaman penambahan kasus uji dapat dilihat pada Gambar 3.20, 3.21 dan 3.22.



Gambar 3.20 Rancangan Antarmuka Halaman Penambahan Kasus Uji Bagian Halaman Keseluruhan

Pada Gambar 3.20, pada tabel yang memuat elemen nomor 4 dan 5, terdapat kolom untuk mengisi data otentikasi yang digunakan. Pada bagian nomor 4 terdapat nama jenis otentikasi, dan pada bagian nomor 5 terdapat kolom teks untuk mengisi data otentikasi. Terdapat 4 jenis pilihan otentikasi yang dapat digunakan, yaitu Open (tanpa otentikasi), Basic, Digest dan Bearer Token. Data-data otentikasi yang terdapat pada tiap-tiap jenis otentikasi dijabarkan pada Tabel 3.19.

Tabel 3.19 Data Otentikasi untuk Tiap Jenis Otentikasi

No	Tipe Otentikasi	Data Otentikasi
1	Open	Tidak ada

2	Basic	-Username -Password
3	Digest	-User -Nonce -Realm -qop -Nonce Count -Client Nonce -Response -Opaque -URI
4	Bearer Token	-Access Token

Bagian nomor 6 berfungsi untuk memilih tipe protocol *request* HTTP yang akan digunakan. Terdapat dua pilihan yang dapat dipilih, yaitu REST dan SOAP.

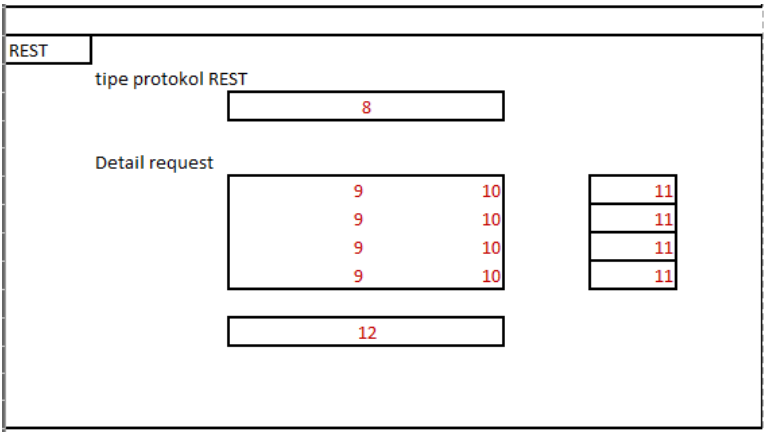
Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.20.

Tabel 3.20 Bagian-Bagian Antarmuka halaman Penambahan Kasus Uji Bagian Halaman Keseluruhan

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Nama kasus uji	*Text input*	Menerima *input* nama kasus uji	Data yang di-*input* pengguna
2	*Flag* publik	*Radio button*	Menerima *input* pilihan apakah kasus uji termasuk public atau tidak	*Mouse click*
3	Pilih tipe otentikasi	*Dropdown*	Menerima *input* pilihan tipe otentikasi	*Mouse Click*
4	Jenis data otentikasi	*text*	Nama jenis data otentikasi	-

5	*Field* data otentikasi	*Text input*	Menerima *input* untuk tiap data jenis otentikasi	Data yang di-*input* pengguna
6	Pilih tipe protokol	*Dropdown*	Menerima *input* pilihan tipe protocol	*Mouse Click*
7	Tombol Simpan	*Button*	Menyimpan data kasus uji	*Mouse Click*

Bagian yang bertanda "data detail kasus uji" berisi bagian halaman antarmuka yang terdapat pada Gambar 3.21 dan 3.22. Salah satu akan terpilih berdasarkan pilihan pada bagian nomor 7.

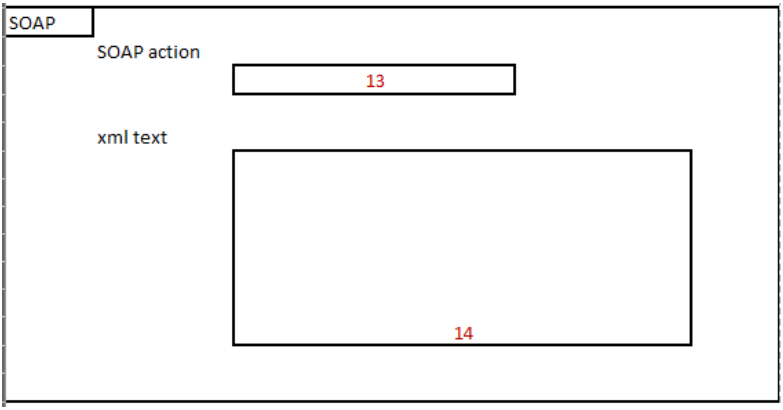


Gambar 3.21 Rancangan Antarmuka halaman Penambahan Kasus Uji Bagian Detail Kasus Uji REST

Penjelasan mengenai bagian-bagian antarmuka pada gambar 3.21 (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.21.

Tabel 3.21 Bagian-Bagian Antarmuka halaman Penambahan Kasus Uji Bagian Detail Kasus Uji REST

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
8	Pilih tipe REST *request*	*Dropdown*	Menerima *input* pilihan tipe REST *request.* Terdapat dua pilihan, yaitu GET dan POST	*Mouse Click*
9	*Field* nama parameter	*Text input*	Menerima *input* nama parameter	Data yang di-*input* pengguna
10	*Field* isi parameter	*Text input*	Menerima *input* isi parameter	Data yang di-*input* pengguna
11	Hapus parameter	*Button*	Hapus parameter yang terdapat di sebelahnya	*Mouse Click*
12	Tambah parameter	*Button*	Menambah Parameter	*Mouse Click*



Gambar 3.22 Rancangan Antarmuka halaman Penambahan Kasus Uji Bagian Detail Kasus Uji SOAP

Penjelasan mengenai bagian-bagian antarmuka pada gambar 3.22 (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.22.

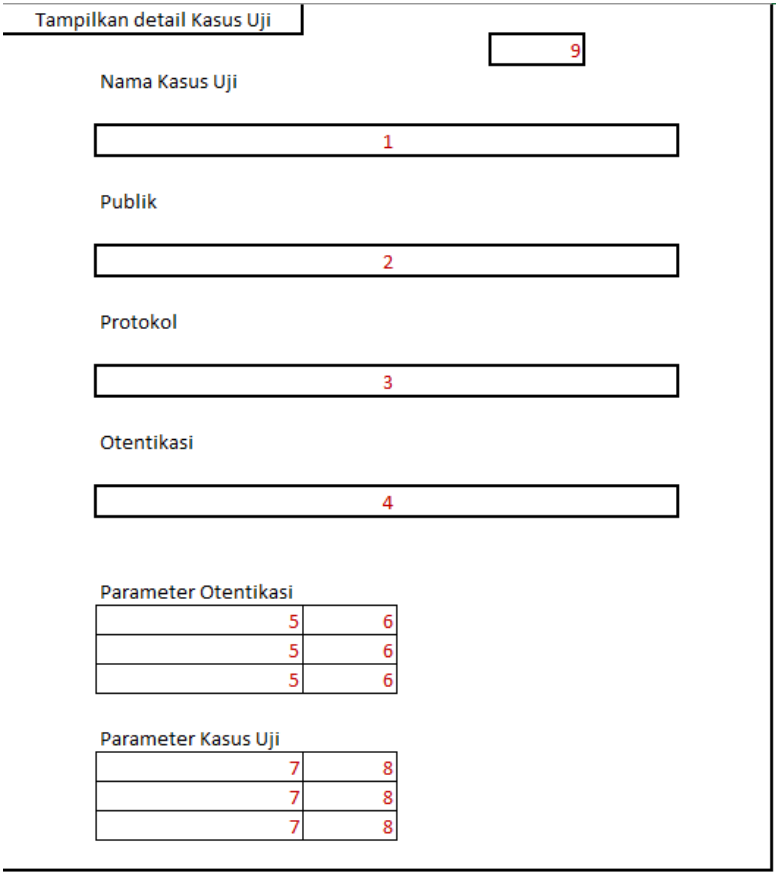
Tabel 3.22 Bagian-Bagian Antarmuka Halaman Penambahan Kasus Uji Bagian Detail Kasus Uji SOAP

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
13	*Field* SOAP Action	*Text input*	Menerima *input* SOAP Action	Data yang di-*input* pengguna
14	*Field* xml text	*Text input*	Menerima *input raw* XML body dari pengguna.	Data yang di-*input* pengguna

3.2.2.4. Antarmuka Halaman Lihat Detail Kasus Uji

Antarmuka halaman lihat detail kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk melihat data kasus

uji untuk satu kasus uji yang spesifik. Rancangan antarmuka halaman ini dapat dilihat pada Gambar 3.23.



Gambar 3.23 Rancangan Antarmuka Halaman Lihat Detail Kasus Uji

Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.23.

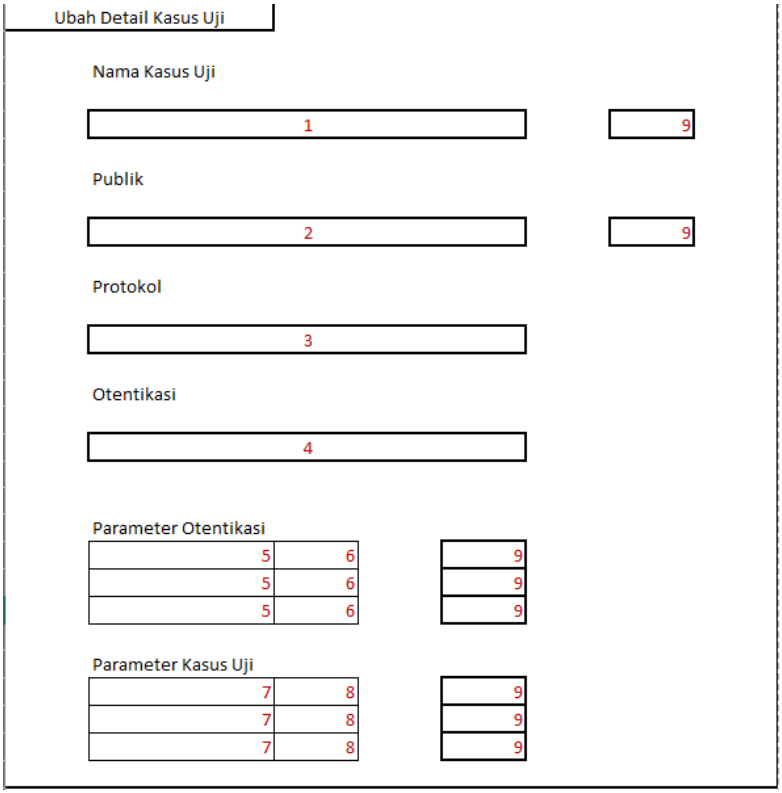
Tabel 3.23 Bagian-Bagian Antarmuka Halaman Lihat Detail Kasus Uji

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Nama kasus uji	*Text*	nama kasus uji	-
2	*Flag* publik	*Text*	Keterangan apakah kasus uji termasuk public atau tidak	-
3	Jenis protokol	*text*	Jenis protocol *request* yang digunakan. Terdapat 3 jenis protokol yang mungkin muncul, yaitu GET, POST dan SOAP	*Mouse Click*
4	Jenis otentikasi	*text*	Nama jenis data otentikasi	-
5	Jenis data otentikasi	*Text*	jenis data otentikasi	-
6	*Field* data otentikasi	*text*	Isi data otentikasi	-
7	Jenis atribut request	*Text*	jenis atribut request	-
8	*Field* data otentikasi	*text*	Isi atribut request	-

Pada bagian 5, 6, 7 dan 8, bentuk bisa bervariasi mengikuti keterangan yang ada.

3.2.2.5. Antarmuka Halaman Ubah Detail Kasus Uji

Antarmuka halaman ubah detail kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk melakukan perubahan pada data kasus uji untuk satu kasus uji yang spesifik. Rancangan antarmuka halaman ini dapat dilihat pada Gambar 3.24.



Gambar 3.24 Rancangan Antarmuka Halaman Ubah Detail Kasus Uji

Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.24.

Tabel 3.24 Bagian-Bagian Antarmuka Halaman Ubah Detail Kasus Uji

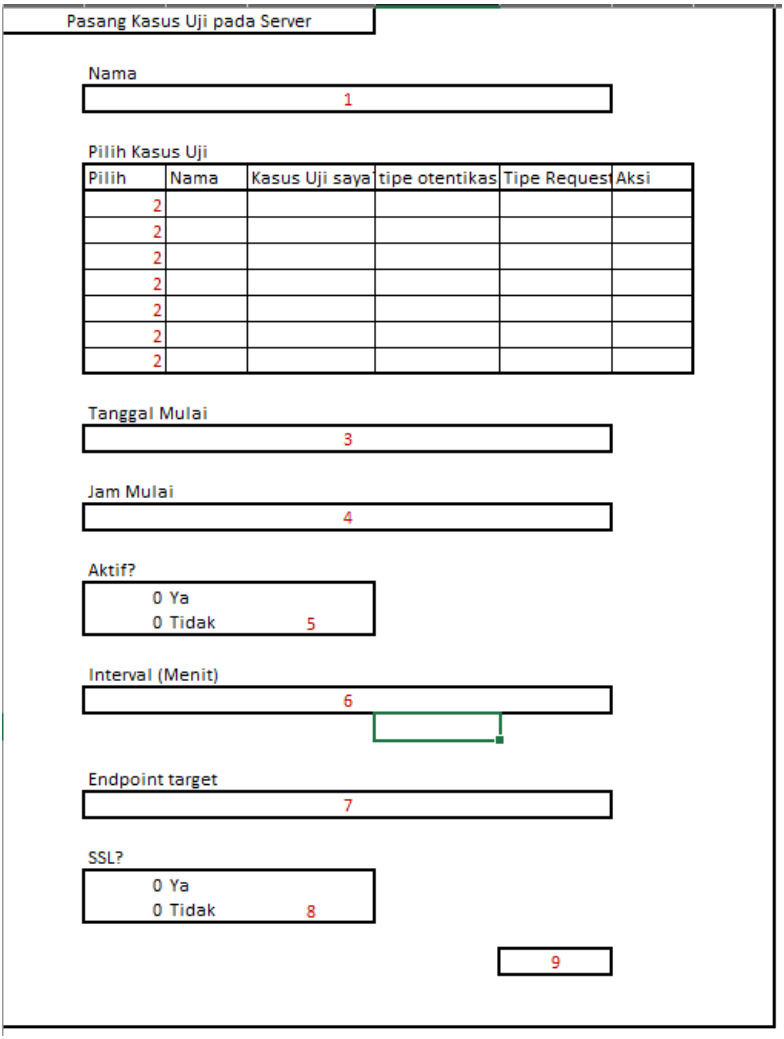
No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Nama kasus uji	*Input Text*	nama kasus uji	Data yang di-*input* pengguna
2	*Flag* publik	*Input Text*	Keterangan apakah kasus uji termasuk public atau tidak	Data yang di-*input* pengguna
3	Jenis protokol	*text*	Jenis protocol *request* yang digunakan. Terdapat 3 jenis protokol yang mungkin muncul, yaitu GET, POST dan SOAP	-
4	Jenis otentikasi	*text*	Nama jenis data otentikasi	-
5	Jenis data otentikasi	*Input text*	jenis data otentikasi	Data yang di-*input* pengguna
6	*Field* data otentikasi	*Input text*	Isi data otentikasi	Data yang di-*input* pengguna
7	Jenis atribut request	*Input text*	jenis atribut *request*	Data yang di-*input* pengguna
8	*Field* data otentikasi	*Input text*	Isi atribut *request*	Data yang di-

				input pengguna
9	Tombol Simpan	*button*	Tombol untuk menyimpan perupahan	*Mouse click*

Pada bagian 5, 6, 7 dan 8, bentuk bisa bervariasi mengikuti keterangan yang ada.

3.2.2.6. Antarmuka Halaman Penambahan Pemasangan Kasus Uji Pada Server

Antarmuka halaman ubah detail kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk melakukan penambahan pemasangan kasus uji pada sebuah server uji. Rancangan antarmuka halaman ini dapat dilihat pada Gambar 3.25.



Gambar 3.25 Rancangan Antarmuka Halaman Penambahan Pemasangan Kasus Uji Pada Server

Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.25.

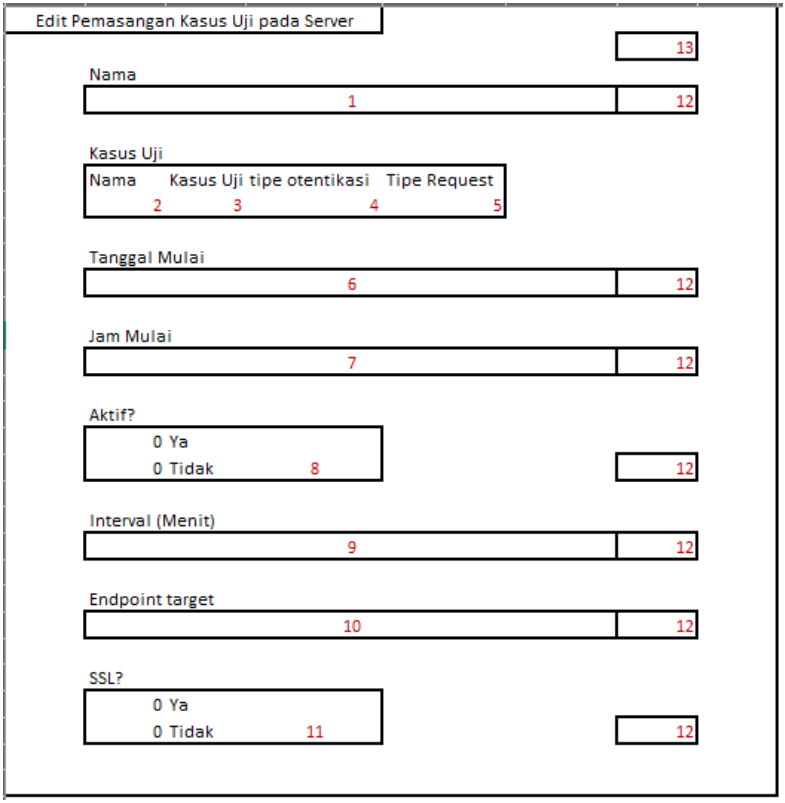
Tabel 3.25 Bagian-Bagian Antarmuka Halaman Penambahan Pemsangan Kasus Uji Pada Server

No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Nama pemasangan kasus uji	*Input Text*	*Input* nama pemasangan kasus uji	Data yang di-*input* pengguna
2	Pilih kasus uji	*Input Text*	Memilih kasus uji yang akan digunakan	Data yang di-*input* pengguna
3	*Input* tanggal mulai	*Datepicker*	Memilih tanggal awal eksekusi pemasangan pengujian	*Mouse click*
4	*Input* waktu mulai	*Timepicker*	Memilih jam awal eksekusi pemasangan pengujian	*Mouse click*
5	Pilih status aktif	*Radio button*	Memilih apakah pemsangan pengujian akan langsung aktif atau tidak	*Mouse click*
6	*Input Interval* pengujian	*Input text*	*Input* waktu interval pengujian (dalam menit)	Data yang di-*input* pengguna
7	*Input endpoint target*	*Input text*	*Input endpoint target server* yang akan digunakan	Data yang di-*input* pengguna
8	Pilih status SSL	*Radio button*	Memilih apakah *endpoint target* akan diakses	*Mouse click*

			menggunakan SSL atau tidak	
9	Tombol Simpan	*button*	Tombol untuk menyimpan data yang dimasukkan	*Mouse click*

3.2.2.7. Antarmuka Halaman Ubah Detail Pemasangan Kasus Uji Pada Server

Antarmuka halaman ubah detail kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk melakukan perubahan detail pemasangan kasus uji yang telah ada sebelumnya. Rancangan antarmuka halaman ini dapat dilihat pada Gambar.



Gambar 3.26 Rancangan Antarmuka Halaman Ubah Detail Pemasangan Kasus Uji Pada Server

Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.26.

Tabel 3.26 Bagian-Bagian Antarmuka Halaman Ubah Detail Pemsangan Kasus Uji Pada Server

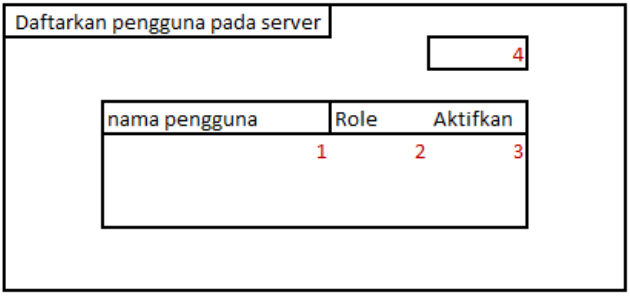
No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran

1	Nama pemasangan kasus uji	*Input Text*	*Input* nama pemasangan kasus uji	Data yang di-*input* pengguna
2	*Nama Kasus Uji*	*Text*	Keterangan apakah kasus uji termasuk public atau tidak	Data yang di-*input* pengguna
3	Kasus Uji saya?	*Text*	Jenis otentikasi *request* yang digunakan pada kasus uji	-
4	Jenis otentikasi	*Text*	Nama jenis data otentikasi	-
5	Tipe Request	*Text*	Jenis protocol *request* yang digunakan. Terdapat 3 jenis protocol yang mungkin muncul, yaitu GET, POST dan SOAP	-
6	*Input* tanggal mulai	*Datepicker*	Memilih tanggal awal eksekusi pemasangan pengujian	*Mouse click*
7	*Input* waktu mulai	*Timepicker*	Memilih jam awal eksekusi pemasangan pengujian	*Mouse click*
8	Pilih status aktif	*Radio button*	Memilih apakah pemsangan pengujian akan langsung aktif atau tidak	*Mouse click*

9	*Input Interval* pengujian	*Input text*	*Input* waktu interval pengujian (dalam menit)	Data yang di-*input* pengguna
10	*Input endpoint target*	*Input text*	*Input endpoint target server* yang akan digunakan	Data yang di-*input* pengguna
11	Pilih status SSL	*Radio button*	Memilih apakah *endpoint target* akan diakses menggunakan SSL atau tidak	*Mouse click*
12	Tombol Simpan	*button*	Tombol untuk menyimpan data yang dimasukkan	*Mouse click*
13	Tombol batal	button	Tombol untuk keluar dari *mode* pengubahan detail pemasangan kasus uji	*Mouse click*

3.2.2.8. Antarmuka Halaman Pemberian Akses Pengguna ke Server

Antarmuka halaman ubah detail kasus uji merupakan antarmuka yang berfungsi sebagai sarana untuk melakukan penambahan pemasangan kasus uji pada sebuah server uji. Rancangan antarmuka halaman ini dapat dilihat pada Gambar 3.27.



Gambar 3.27 Rancangan Antarmuka Halaman Pemberian Akses Pengguna ke Server

Penjelasan mengenai bagian-bagian antarmuka (ditandai dengan angka berwarna merah) terdapat pada Tabel 3.27.

Tabel 3.27 Bagian-Bagian Antarmuka Halaman Pemberian Akses Pengguna ke Server

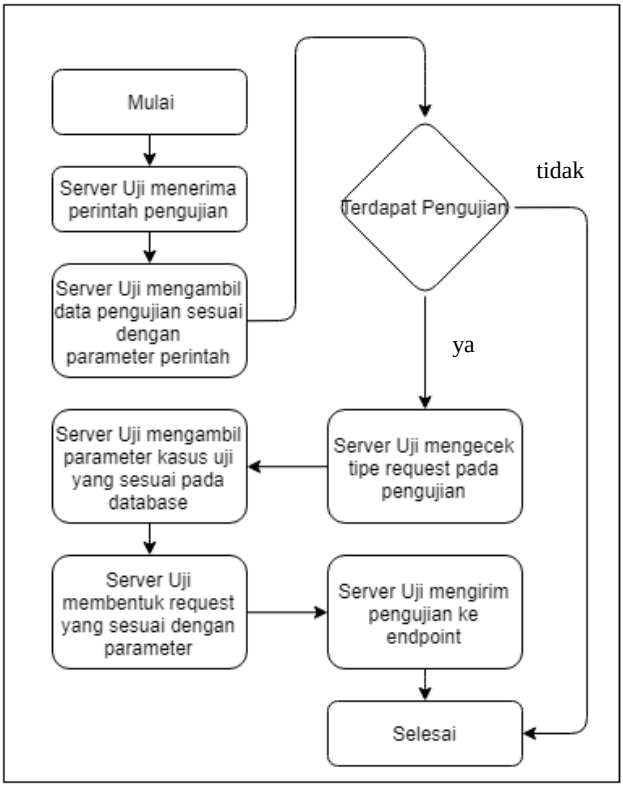
No	Nama Bagian Antarmuka	Jenis	Kegunaan	Masukan/ Keluaran
1	Nama pengguna	*Text*	Nama Pengguna	-
2	*Role*	*Text*	Peran pengguna pada sistem	-
3	Aktifkan	*Button*	Tombol yang berfungsi sebagai *toggle* aktif dan tidak aktif	*Mouse Click*
4	Pilih Server	*Dropdown*	Pemilihan server uji	*Mouse Click*

3.2.3. Perancangan Proses

Pada Subbab ini akan dijelaskan mengenai perancangan proses-proses yang ada pada perangkat lunak. Berikut ini merupakan rancangan proses-proses yang ada pada pengembangan Tugas Akhir ini.

3.2.3.1. Proses Pada Kasus Penggunaan Mengirim *Request* Pada *Endpoint* Sesuai Kasus Uji

Proses yang terdapat pada spesifikasi kasus penggunaan Mengirim *Request* pada *Endpoint* Sesuai Kasus Uji, digambarkan pada gambar 3.28.

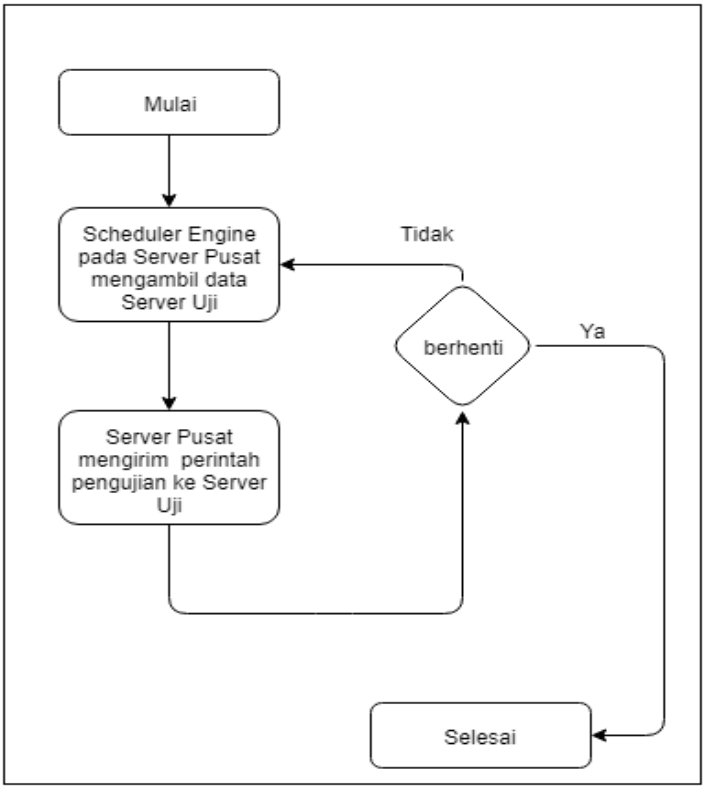


Gambar 3.28 Proses pada Kasus Penggunaan Mengirim *Request* pada *Endpoint* sesuai Kasus Uji

Proses ini terdapat pada perangkat lunak *Server* Uji. Pada proses ini, pengiriman pengujian ke *target server* dieksekusi. Proses ini diinisiasi dengan perintah pengujian dari perangkat lunak *Server* Pusat.

3.2.3.2. Proses Pada Kasus Penggunaan Mengirimkan Perintah Pengujian ke *Server* Uji

Proses yang terdapat pada spesifikasi kasus penggunaan Mengirim perintah pengujian ke Server Uji digambarkan pada gambar 3.29.



Gambar 3.29 Proses pada Kasus Penggunaan Mengirim *Request* pada *Endpoint* sesuai Kasus Uji

Proses ini dilakukan dengan *loop* terus-menerus setiap menitnya oleh *scheduler engine* yang ada pada *server* pusat hingga *scheduler engine* dihentikan.

3.2.4. Perancangan Daftar Perintah

Terdapat dua perintah pada aktor *Server* Uji dan *Server* Pusat. Satu perintah terdapat pada *Server* Pusat, dan satu perintah pada *Server* Uji. *Service* pada masing-masing *Server* Pusat dan *Server* Uji dapat diakses menggunakan *endpoint* yang diakses menggunakan metode GET. Deskripsi perintah pada masing-masing *server* beserta sintaks perintah terdapat pada tabel 3.28 dan 3.29.

Tabel 3.28 Daftar Perintah pada *Server* Pusat

Deskripsi Perintah	Sintaks Perintah
Perintah untuk menyalakan *Scheduler* pada *Server* Pusat	/start

Tabel 3.29 Daftar Perintah pada *Server* Uji

Deskripsi Perintah	Sintaks Perintah
Perintah untuk mencari dan mengeksekusi pengujian Mempunyai parameter (*time*, *address*)	/entrypoint?time=(*time*)&address=(*address*)

3.2.5. Perancangan Basis Data

Rancangan tabel-tabel beserta deskripsi atribut pada basis data yang akan digunakan pada sistem akan dibahas pada subbab-subbab berikut ini. *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM) dari basis data terdapat pada lampiran A dan B.

3.2.5.1. Tabel *User*

Tabel *user* digunakan untuk menyimpan daftar pengguna yang dapat mengakses sistem, berserta perannya dalam sistem. Detail atribut tabel ini terdapat pada tabel 3.30.

Tabel 3.30 Rancangan Tabel *User*

Atribut	Tipe Data	Deskripsi

Id_user	Integer	*Primary Key*
Username	Varchar(60)	Nama pengguna
Password	Varchar(60)	Password pengguna
Role	Varchar(10)	Peran pengguna dalam sistem

3.2.5.2. Tabel *Server*

Tabel *server* digunakan untuk menyimpan data *server* yang terdaftar pada sistem. Detail atribut tabel ini terdapat pada tabel 3.31.

Tabel 3.31 Rancangan Tabel *Server*

Atribut	Tipe Data	Deskripsi
Id_server	Integer	*Primary Key*
Address	Varchar(20)	Alamat server
Name	Varchar(30)	Nama server
Active	Numeric(1,0)	Status aktif server (1 atau 0)

3.2.5.3. Tabel *Server User*

Tabel server *user* digunakan untuk menyimpan data pengguna mana sajakah yang mendapatkan akses untuk memasang kasus uji pada sebuah server. Detail atribut tabel ini terdapat pada tabel 3.32.

Tabel 3.32 Rancangan Tabel *Server User*

Atribut	Tipe Data	Deskripsi
Id_server_user	Integer	*Primary Key*
Id_server	Integer	*Foreign Key* tabel *server*
Id_user	Integer	*Foreign Key* tabel *user*

3.2.5.4. Tabel *Testcase*

Tabel *testcase* digunakan untuk menyimpan data kasus uji yang ada pada sistem. Detail atribut tabel ini terdapat pada tabel 3.33.

Tabel 3.33 Rancangan Tabel *Testcase*

Atribut	Tipe Data	Deskripsi
Id_testcase	Integer	*Primary Key*
Id_request_type	Integer	*Foreign Key* tabel *request type*
Id_user	Integer	*Foreign Key* tabel *user*
Testcase_name	Varchar(100)	Nama kasus uji
Is_public	Numeric(1,0)	*Flag* penanda apakah kasus uji bisa diakses oleh pengguna lainnya
Soft_delete	Numeric (1,0)	*Flag* apakah kasus uji aktif atau tidak

3.2.5.5. Tabel *Server Testcase*

Tabel *server testcase* digunakan untuk menyimpan data pemasangan kasus uji pada server beserta data-data pemasangan kasus uji tersebut. Detail atribut tabel ini terdapat pada tabel 3.34.

Tabel 3.34 Rancangan Tabel *Server Testcase*

Atribut	Tipe Data	Deskripsi
Id_server_testcase	Integer	*Primary Key*
Id_server	Integer	*Foreign Key* tabel *server*
Id_testcase	Integer	*Foreign Key* tabel *request type*
Id_user	Integer	*Foreign Key* tabel *request type*
Active	Integer	Status aktif pemasangan kasus uji
Next_test	Datetime	Waktu pengujian berikutnya
Interval	Integer	Jarak antar pengujian
Endpoint_target	Varchar(100)	Target *endpoint* pengujian
Is_secure	Integer	*Flag* penanda apakah ssl akan digunakan

Testcase_server_name	Text	Nama pemasangan kasus pengujian

3.2.5.6. Tabel *Auth*

Tabel *user* digunakan untuk menyimpan daftar pengguna yang dapat mengakses sistem, berserta perannya dalam sistem. Detail atribut tabel user terdapat pada tabel 3.35.

Tabel 3.35 Rancangan Tabel *Auth*

Atribut	Tipe Data	Deskripsi
Id_auth	Integer	*Primary Key*
Auth_type	Varchar(60)	Tipe otentikasi yang terdaftar

3.2.5.7. Tabel *Testcase Auth*

Tabel *testcase auth* digunakan untuk menyimpan data otentikasi pemasangan kasus uji. Detail atribut tabel ini terdapat pada tabel 3.36.

Tabel 3.36 Rancangan Tabel *Testcase Auth*

Atribut	Tipe Data	Deskripsi
Id_req_auth	Integer	*Primary Key*
Id_testcase	Integer	*Foreign Key* tabel *request type*
Id_auth	Integer	*Foreign Key* tabel *request type*

3.2.5.8. Tabel *Server Testcase Result*

Tabel *server testcase result* digunakan untuk menyimpan hasil pengujian dari pemasangan kasus uji. Detail atribut tabel ini terdapat pada tabel 3.37.

Tabel 3.37 Rancangan Tabel *Server Testcase Result*

Atribut	Tipe Data	Deskripsi
Id_test_result	Integer	*Primary Key*

Id_server_testcase	Integer	*Foreign Key* tabel *server testcase*
Response_time	Integer	Waktu respon pengujian
Response_code	Integer	Kode respon
Date_created	Datetime	Waktu eksekusi

3.2.5.9. Tabel *Testcase Detail*

Tabel *testcase detail* digunakan untuk menyimpan atribut-atribut dari kasus uji. Detail atribut tabel ini terdapat pada tabel 3.38.

Tabel 3.38 Rancangan Tabel *Testcase Detail*

Atribut	Tipe Data	Deskripsi
Id_request_detail	Integer	*Primary Key*
Id_testcase	Integer	*Foreign Key* tabel *request type*
Request_detail_key	Varchar(100)	Nama atribut *request*
Request_detail_value	Text	Isi atribut *request*

3.2.5.10. Tabel *Auth Type Fields*

Tabel *auth type fields* digunakan untuk menyimpan *template* atribut-atribut dari tiap jenis otentikasi pada sistem. Detail atribut tabel ini terdapat pada tabel 3.39.

Tabel 3.39 Rancangan Tabel *Auth Type Fields*

Atribut	Tipe Data	Deskripsi
Id_auth_type_fields	Integer	*Primary Key*
Id_auth	Integer	*Foreign Key* tabel *request type*
Field_name	Varchar(30)	Nama field dari otentikasi

3.2.5.11. Tabel *Auth Detail Value*

Tabel *auth detail value* digunakan untuk menyimpan data atribut otentikasi milik sebuah kasus uji sesuai dengan template

pada tabel *auth type fields*. Detail atribut tabel ini terdapat pada tabel 3.40.

Tabel 3.40 Rancangan Tabel *Auth Detail Value*

Atribut	Tipe Data	Deskripsi
Id_auth_detail_value	Integer	*Primary Key*
Id_auth_type_fields	Integer	*Foreign Key* tabel *request type*
Id_req_auth	Integer	*Foreign Key* tabel *request type*
Value	Varchar(100)	Isi field otentikasi

3.2.5.12. Tabel *Request Type*

Tabel *request type* digunakan untuk menyimpan daftar jenis *request* yang dapat dibuat oleh sistem. Detail atribut tabel ini terdapat pada tabel 3.41.

Tabel 3.41 Rancangan Tabel *Request Type*

Atribut	Tipe Data	Deskripsi
Id_request_type	Integer	*Primary Key*
Id_protocol	Integer	*Foreign Key* tabel *request type*
Request_type_name	Varchar(60)	Tipe request yang terdaftar pada sistem

3.2.5.13. Tabel *Protocol*

Tabel *protocol* digunakan untuk menyimpan daftar protocol *request* HTTP yang dapat dibuat oleh sistem. Detail atribut tabel ini terdapat pada tabel 3.42.

Tabel 3.42 Rancangan Tabel *Protocol*

Atribut	Tipe Data	Deskripsi
Id_protocol	Integer	*Primary Key*
Protocol_name	Varchar(10)	Nama protokol

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Terdapat dua Bahasa pemrograman yang digunakan dalam implementasi system, yaitu Bahasa Pemorgraman Java dan PHP.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Sistem

Perangkat	Spesifikasi
Perangkat keras	*Server* Pusat: Prosesor: Intel(R) Core i7 7700HQ (4 cores @ 2.80 GHz) Memori: 16384MB *Server* Uji: Prosesor: Intel(R) Core i7 7500U (2 cores @ 1.80 GHz) Memori: 4096MB
Perangkat lunak	Sistem Operasi (*Server* Pusat): Microsoft Windows 10 SL 64-bit Sistem Operasi (*Server* Uji): Microsoft Windows 10 Pro 64 bit DBMS: SQL *Server* 2012R2 *Server*: Tomcat 7.0, XAMPP Perangkat Pengembang: Eclipse Kepler R2, Sublime Text 3 Perangkat Pembantu: Microsoft Word 2016

Wifi	*Mifi* Huawei E5577

4.2. Deskripsi Umum dan Batasan Implementasi

Pada subbab ini akan dibahas mengenai deskripsi umum pada implementasi tugas akhir, serta batasan-batasan dan spesifikasi yang ditentukan pada sistem.

4.2.1. Deskripsi Umum Implementasi

Pada implementasi tugas akhir ini, terdapat tiga bagian perangkat lunak sesuai dengan perancangan aplikasi bagian arsitektur sistem. Ketiga bagian ini masing-masing adalah perangkat lunak pada *Server* Uji, perangkat lunak pada *Server* Pusat dan Aplikasi *Dashboard.*

Bagian perangkat lunak pada *Server* Uji berfungsi untuk melakukan eksekusi pengujian dengan pemasangan kasus uji. Inisiasi eksekusi pengujian dilakukan oleh *Server* Pusat. Perangkat lunak ini selanjutnya disebut dengan *Request Engine.*

Bagian perangkat lunak pada *Server* Pusat berfungsi untuk menjalankan fungsi *scheduler* serta pengirim perintah pengujian pada *Request Engine*. Perangkat lunak ini selanjutnya disebut *Scheduler Engine.*

Bagian Aplikasi *Dashboard* adalah aplikasi berbasis *web* yang memiliki fungsi-fungsi yang terdapat pada bab 3 bagian perancangan kasus penggunaan.

4.2.2. Batasan dan Spesifikasi Tipe *Request* Pada Kasus Uji

Pada *Request Engine,* terdapat proses pengujian kasus uji. Pembuatan kasus uji dibatasi pada tiga tipe *request*, yaitu *request* GET, POST dan SOAP. Spesifikasi masing-masing tipe *request* yang dapat dieksekusi pada sistem ini terdapat pada tabel 4.2, 4.3 dan 4.4.

Tabel 4.2 Tabel Spesifikasi Kasus Uji Dengan Tipe GET

Tipe protokol	GET
Tipe data parameter *request*	*Text*

Struktur parameter	*Key* dan *Value*
Contoh parameter	*Key1 = ”Address”* *Value1=”192.168.0.1”* *Key2 = “time”* *Value2 =”10/05/2018 13:00:00*
Peletakan parameter	Pada URL *request*
Contoh peletakan parameter	*“{Server Address}* /entrypoint?Address=(*time*)& time=10/05/2018 13:00:00

Tabel 4.3 Tabel Spesifikasi Kasus Uji Dengan Tipe POST

Tipe protokol	POST
Tipe data parameter *request*	*Text*
Struktur parameter	*Key* dan *Value*
Contoh parameter	*Key1 = ”Address”* *Value1=”192.168.0.1”* *Key2 = “time”* *Value2 =”10/05/2018 13:00:00*
Peletakan parameter	Pada *body request*
Contoh peletakan parameter	Address=192.168.0.1&time=10/05/2018 13:00:00

Tabel 4.4 Tabel Spesifikasi Kasus Uji Dengan Tipe SOAP

Tipe protokol	SOAP
Tipe data parameter *request*	*Text*
Struktur parameter	SOAP Action dan XML *Text*
Contoh parameter	SOAP Action = http://tempuri.org/Add XML text = <?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

	xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <Add xmlns="http://tempuri.org/"> <intA>3</intA> <intB>4</intB> </Add> </soap:Body> </soap:Envelope>
Peletakan parameter	-SOAP Action diletakkan pada HTTP *Header* -XML *Text* Diletakkan pada *body request*
Contoh peletakan parameter	*//request header* SOAP Action: http://tempuri.org/Add *//request body* <?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <Add xmlns="http://tempuri.org/"> <intA>3</intA> <intB>4</intB> </Add> </soap:Body> </soap:Envelope>

4.2.3. Batasan dan Spesifikasi Tipe Otentikasi Pada Kasus Uji

Metode Otentikasi HTTP yang dapat digunakan pada kasus uji dibatasi pada tiga tipe otentikasi, yaitu *Basic Authentication, Digest Authentication* dan *Bearer Token.* Spesifikasi masing-masing metode otentikasi terdapat pada tabel 4.5,4.6 dan 4.7.

Tabel 4.5 Spesifikasi Metode Otentikasi *Basic Authentication*

Tipe protokol	*Basic Authentication*
Tipe data parameter *request*	*Text*
Struktur parameter	*Username dan Password*
Contoh parameter	Username=”postman” Passsword=”password”
Peletakan parameter	Pada HTTP *Header*, dengan dikonversi terlebih dahulu menjadi token base64 dengan struktur: username:password
Contoh peletakan parameter	Authentication= Basic cG9zdG1hbjpwYXNzd29yZA==

Tabel 4.6 Spesifikasi Metode Otentikasi *Digest Authentication*

Tipe protokol	*Digest Authentication*
Tipe data parameter *request*	*Text*
Struktur parameter	-Username -Nonce -Realm -qop -Nonce Count -Algorithm -Client Nonce -Response -Opaque -URI

Contoh parameter	username="postman", realm="Users", nonce="LztT2EMJfBeNfZgJW7oLT6HWG4Y9wxdc", uri="/digest-auth", algorithm="MD5-sess", qop=auth, nc=00000001, cnonce="asdasdasdasd", response="debeac466c7414c830d28bf20f626431"
Peletakan parameter	Pada HTTP Header, diletakkan di sebelah nama jenis otentikasi
Contoh peletakan parameter	Authentication= Digest Username="postman", realm="Users", nonce="ni1LiL0O37PRRhofWdCLmwFsnEtH1lew", uri="/digest-auth", response="254679099562cf07df9b6f5d8d15db44", opaque=""

Tabel 4.7 Spesifikasi Metode Otentikasi *Bearer Token*

Tipe protokol	*Bearer Token*
Tipe data parameter *request*	*Text*
Struktur parameter	*Bearer Token*
Contoh parameter	Token= Zm9vOmJhcg==
Peletakan parameter	Pada HTTP *Header*, diletakkan di sebelah nama jenis otentikasi
Contoh peletakan parameter	Authentication= Bearer Zm9vOmJhcg==

4.3. Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang telah dijelaskan pada bagian analisis dan perancangan. Proses-proses yang terdapat pada subbab ini telah disesuaikan agar dapat memenuhi spesifikasi kasus penggunaan perangkat lunak yang dibahas pada bab 3 bagian analisis dan perancangan.

4.3.1. Implementasi Tahap Pengiriman *Request*

Subbab ini membahas mengenai implementasi eksekusi kasus uji oleh *Server* Uji. Pembahasan pada subbab ini mencakup proses yang dibutuhkan pada kasus penggunaan SU01 dan SU02 yang telah dibahas pada bab perancangan sistem bagian perancangan kasus penggunaan.

Setelah *Server* Uji mendapatkan perintah pengujian dari *Server* Pusat, *Server* Uji mengeksekusi pengiriman pengujian. Terdapat empat tahap eksekusi pengiriman pengujian, yaitu inisiasi *Request Engine*, pengumpulan kasus uji, pengumpulan detail kasus uji, dan eksekusi kasus uji.

Proses pengiriman pengujian dimulai dari inisiasi perangkat lunak *Scheduler Engine* pada fungsi *entrypoint*. Fungsi ini terdapat dalam *class driver* yang terdapat pada *Request Engine*. Kode semu fungsi ini terdapat pada gambar 4.1.

```
1  Entrypoint(time, address)
2  Ambil data pengujian//(dengan parameter time dan address)
3  if data>0
4      For data as i:
           HandleTestcase (i)
```

Gambar 4.1 Kode Semu Fungsi *Entrypoint*

Deklarasi *endpoint* fungsi *entrypoint* milik *Request Engine* pada server uji ditunjukkan pada kode sumber 4.1.

```
1  @Path("/entrypoint")
2  public class driver {
3         @GET
4          @Produces(MediaType.TEXT_PLAIN)
5          @Consumes(MediaType.TEXT_PLAIN)
6         public String Entrypoint( ("time") String
   time, ("server") String server)
7  throws URISyntaxException,MalformedURLException,
   SOAPException
          {
```

Kode Sumber 4.1 Deklarasi *Endpoint* Fungsi *Entrypoint*

Fungsi *entrypoint* adalah bagian yang paling awal dari perangkat lunak *Request Engine*. Fungsi *entrypoint* diaktifkan dengan cara mengakses *endpoint "{Server Address}* /entrypoint?time=(*time*)& address=(*address*)" dengan metode HTTP GET. Terdapat dua parameter yang diterima oleh fungsi ini, yaitu *time* dan *address*.

Dua parameter ini akan digunakan untuk mengambil pemasangan kasus uji yang sesuai. *Query* yang dibutuhkan untuk mengambil pemasangan kasus uji yang sesuai terdapat pada Kode Sumber 4.2.

```
SELECT st.* FROM [dbo].[testcase] t, [dbo].[server] s,
[dbo].[SERVER_TESTCASE] st
where s.address='{server}'
and
DATEADD (MINUTE, DATEDIFF (MINUTE, 0, '{time}'), 0)
=st.next_test
and
t.id_testcase= st.id_testcase
and
s.id_server= st.id_server
and
st.active=1
```

Kode Sumber 4.2 *Query* untuk Mengambil Kasus Uji

Kriteria kasus uji yang dipilih pada pengambilan dengan *query* pada kode sumber 4.2 adalah pemasangan kasus uji yang

memiliki data *next_test* yang berada pada menit yang sama dengan waktu pada *server* pusat pada saat perintah pengujian dikirimkan, serta memiliki nilai kolom *active*=1.

Jika terdapat pemasangan pengujian yang sesuai, maka pemasangan pengujian tersebut akan dieksekusi. Eksekusi pemasangan pengujian dilakukan oleh fungsi bernama *HandleTestcase*.

Fungsi *HandleTestcase* Terdapat pada kelas *Testcasehandlers*. Fungsi ini menerima satu pemasangan pengujian yang spesifik pada tiap kali fungsi ini dijalankan. Kode semu fungsi ini terdapat pada Gambar 4.2.

```
1  Params = obtainTestcaseDetails () //mengambil data testcase
2  Result=executeTestcase(Params) //eksekusi testcase
3  putBack(Result) //memasukkan data hasil pengujian ke database
```

Gambar 4.2 Kode Semu Fungsi *HandleTestcase*

Terdapat 4 parameter yang dibutuhkan pada pemanggilan fungsi ini oleh fungsi *entrypoint*, yang terdapat pada tabel 4.8.

Tabel 4.8 Parameter Pemanggilan Fungsi *HandleTestcase*

no	Nama parameter	Deskripsi	Tipe data
1	*Testcase_server_id*	*id* pemasangan pengujian. Digunakan pada saat akan memasukkan hasil pengujian	Integer
2	*Testcase_id*	Id kasus uji. Digunakan untuk mengambil data kasus uji	Integer
3	*Endpoint*	*Alamat server* tujuan. Digunakan untuk menginisiasi *url target.*	String
4	*Secure*	Digunakan untuk menginialisasi HTTPS (jika *secure*=1)	integer

Pada fungsi *HandleTestcase*, data pemasangan pengujian yang dimasukkan akan dieksekusi. Terdapat tiga tahap eksekusi pengujian pada fungsi *HandleTestcase*, yaitu pengumpulan detail kasus uji, eksekusi pemasangan kasus uji, dan penyimpanan hasil eksekusi pengujian.

Pengumpulan data pemasangan pengujian pada fungsi *HandleTestcase* dilakukan oleh pemanggilan fungsi *obtainTestcaseDetails*. Parameter yang dibutuhkan pada pemanggilan fungsi ini adalah *id_testcase*. Untuk sebuah *id_testcase* milik kasus uji, terdapat empat jenis parameter yang diambil. Daftar parameter yang diambil terdapat pada tabel 4.9.

Tabel 4.9 Parameter Hasil Fungsi *ObtainTestcaseDetails*

no	Nama parameter	Deskripsi	Tipe Data
1	*Tc_type*	Tipe *request* kasus uji.	String
2	*Request_details*	Digunakan untuk mengambil data kasus uji	Map <String, String>
3	*Auth_type*	Digunakan untuk menginisiasi *url target*	String
4	*Auth_detail*	Digunakan untuk menginialisasi *https* (jika *secure*=1)	Map <String, String>

Parameter-parameter pada tabel 4.9 akan digunakan untuk mengeksekusi pengujian dengan kasus uji. Pada proses ini, terdapat tiga tahap, yaitu pembuatan *auth header*, pembuatan *request*, serta pengiriman dan pengecekan *response* pada pengiriman.

Ketiga proses ini dieksekusi oleh sebuah fungsi bernama *handleRequest*. Fungsi ini terdapat pada *class requesthandlers*. Parameter yang dibutuhkan untuk menjalankan fungsi ini adalah seluruh parameter yang didapatkan dari fungsi *obtainTestcaseDetails*, ditambah dengan parameter *endpoint* dan

secure dari eksekusi fungsi *HandleTestcase*. Kode semu fungsi ini terdapat pada Gambar 4.3.

```
1  handleRequest(tc_type, Request_details, auth_type, auth_detail, endpoint, secure)
2      if secure=1:
           url=”https://”+endpoint
       else
           url=”http://”+endpoint
3      Header= headerFactory.createHeader(auth_type, auth_detail)
4      If tc_type=”GET”:
           Result=this.execute_get(request_details,header,url,secure)
5      If tc_type=”POST”:
           Result=this.execute_post(request_details,header,url,secure)
6      If tc_type=”SOAP”:
           Result=this.execute_soap(request_details,header,url,secure)
7      Return result
```

Gambar 4.3 Kode Semu Fungsi *handleRequest*

Kode baris kedua pada gambar 4.3 berfungsi untuk mengatur agar *url* pada fungsi menjadi sesuai dengan variabel *secure*. Pada baris ketiga, terdapat sebuah fungsi bernama *createHeader*. Fungsi tersebut terdapat pada class *HeaderFactory*. Fungsi ini berguna untuk membuat *String* otentikasi yang akan dipasang pada *request header*. kode semu fungsi ini terdapat pada gambar 4.4.

```
1  createHeader(auth_type, auth_detail)
       if auth_type=”Open”:
           Result=null
2      If auth_type =”Basic”:
           String authorization = toBase64(username+":"+password)
           Result= "Basic "+authorization
3      If auth_type=”Digest”:
           String authorization=””
           For i in auth_detail:
```

	authorization= authorization+ " "+ key + "=" +value+" Result="Digest "+ authorization
4	If auth_type ="Bearer": Result= "Bearer "+ token
5	Return result

Gambar 4.4 Kode Semu Fungsi createHeader

Parameter yang diperlukan dalam pemanggilan fungsi ini yaitu *auth_type* dan *auth_detail*. Struktur data variabel *auth_detail* telah dijelaskan pada bab 4 bagian Batasan implementasi.

Hasil fungsi *createHeader* kemudian dikembalikan pada fungsi *handleRequest* untuk dipasang pada HTTP *header*. Implementasi pemrograman untuk memasang parameter otentikasi pada HTTP *header* terdapat pada kode sumber 4.3.

1	conn.setRequestProperty ("Authorization", header);

Kode Sumber 4.3 Pemasangan parameter otentikasi pada HTTP

Pada gambar 4.3 poin nomor 4, 5 dan 6, terdapat tiga fungsi, yaitu execute_get, execute_post, dan execute_soap. Secara struktur, ketiga fungsi ini memiliki alur kerja yang sama, sehingga dapat direpresentasikan dengan kode semu yang sama. Selain itu, ketiga fungsi ini memiliki parameter fungsi yang sama, hanya *array* dalam variabel tersebut memiliki konten yang berbeda. Kode semu yang dapat merepresentasikan ketiga fungsi tersebut terdapat pada Gambar 4.5.

1	Execute()
2	appendData()
3	appendAuth()
4	setRequestmethod()
5	startTimer()
6	connect()
7	responseCode=getResponseCode()
8	ResponseTime=countTime() Return responseCode,responseTime

Gambar 4.5 Kode Semu untuk Fungsi *Execute_get, Execute_post* dan *Execute_soap*

Bagian *appendData* befungsi untuk memasukkan detail kasus uji pada eksekusi pengujian. Bagian *appendAuth* berfungsi untuk memasukkan data otentikasi pada *request header*. Bagian *setRequestMethod* berfungsi untuk memasukkan tipe *request* pada eksekusi. Bagian nomor 5 hingga 8 berfungsi untuk mengeksekusi dan mengambil data *response time* serta *response code* dari pengujian. Data ini kemudian akan dikembalikan ke fungsi *handleTestcase* untuk kemudian dimasukkan ke dalam sistem.

Perbedaaan ketiga fungsi ini dalam bagian-bagian kode semu pada gambar 4.5 terdapat pada tabel 4.10.

Tabel 4.10 Perbedaan Fungsi *Execute_get, Execute_post dan Execute_soap*

No	Bagian	GET	POST	SOAP
1	Letak appendData	Penambahan data dilakukan pada URL (dengan separator &)	Penambahan data dilakukan pada *request body*	Penambahan data pada request dilakukan pada bagian *body* (berupa XML text) dan pada *header* (berupa SOAP *Action*)
2	Keberadaan bagian appendData	opsional	wajib	wajib
3	AppendAuth	identik	identik	identik

Pada Implementasi pemrograman, ketiga fungsi ini dieksekusi dengan *library* yang sama, yaitu dengan *library HTTPURLConnection*. Library ini tersedia pada bahasa pemrograman Java.

Parameter tipe *request* (GET, POST) perlu dipasang pada eksekusi pengiriman *request* sebelum dikirim. Implementasi pemasangan parameter tipe *request* terdapat pada kode sumber 4.4.

```
1  conn.setRequestMethod("{tipe request}");
```

Kode Sumber 4.4 Pemasangan Parameter tipe *request*

Untuk tipe *request* GET, parameter yang dimasukkan adalah "GET". Sedangkan untuk tipe POST dan SOAP, parameter yang dimasukkan adalah "POST"

Selain parameter tipe *request*, hal lain yang perlu dipasang yaitu detail *request*. Terdapat metode pemasngan detail *request* yang berbeda-beda, tergantung pada jenis *request* yang dibuat.

Parameter detail *request* GET dimasukkan ke dalam *request* yang dibuat sesuai dengan struktur yang terdapat pada tabel 4.2. kemudian parameter diletakkan pada URL sebelum *request* dikirim.

Parameter detail *request* POST dimasukkan ke dalam *request* yang dibuat sesuai dengan struktur yang terdapat pada tabel 4.3. parameter-parameter ini dimasukkan pada *body request* sebelum *request* dikirim.

Parameter detail *request* SOAP dimasukkan ke dalam *request* yang dibuat sesuai dengan struktur yang terdapat pada tabel 4.4. parameter SOAP *Action* dimasukkan pada HTTP *header*, sementara XML *text* dipasang pada *body request*.

Implementasi pemrograman untuk melakukan pemasangan parameter pada HTTP *header* untuk SOAP request dan pemasangan *body request* untuk *request* POST dan SOAP masing-masing terdapat pada kode sumber 4.5 dan 4.6.

```
1  conn.setRequestProperty("SOAPAction",soap_action);
```

Kode Sumber 4.5 Pemasangan Parameter Pada HTTP *header* untuk SOAP request

```
1  conn.getOutputStream().write(DataBytes);
```

Kode Sumber 4.6 Pemasangan Parameter Pada *Body Request*

Setelah parameter-parameter dipasang pada *request*, pengiriman request ke *target server* akan dilakukan. Waktu pengiriman request akan dihitung dengan fungsi *timer*. *Response code* yang dikembalikan oleh sistem akan diambil dan disimpan sebagai data pengujian. Bagian implementasi program untuk mengeksekusi pengiriman *request*, penghitungan waktu eksekusi dan pengambilan hasil *response code* terdapat pada kode sumber 4.7.

```
1  long startTime = System.currentTimeMillis();
2  conn.connect();
3  long elapsedTime = System.currentTimeMillis() - startTime;
4  time=(int)elapsedTime;
5  response=conn.getResponseCode();
```

Kode Sumber 4.7 Pengiriman *Request* dan Pengambilan Data *Response Code* dan *Response Time*

Data *response time* dan *response code* yang dihasilkan dari potongan kode sumber 4.7 kemudian dikembalikan oleh fungsi *handleRequest* kepada fungsi *handleTestcase* untuk kemudian disimpan pada sistem. Fungsi yang bertugas untuk melakukan proses ini yaitu fungsi *putBack* yang terdapat pada *class testcaseHandler*. Parameter yang diperlukan pada fungsi ini adalah *id_server_testcase*, *response time* dan *response code*. Kode semu fungsi ini terdapat pada gambar 4.6.

```
1  putBack(responseCode,responseTime,testcase_server_id)
2    insertResult(testcase_server_id, responseTime,responseCode)
   //menyimpan data pengujian
3    updateNextTest(testcase_server_id)
     //perbarui waktu pengujian terakhir
```

Gambar 4.6 Kode Semu Fungsi putBack

Pada fungsi *putBack*, data hasil pengujian pemasangan kasus uji dimasukkan ke tabel *server_testcase_result*. Data yang disimpan adalah data waktu pengujian, *response time* dan *response code*.

Setelah pengujian dilakukan, maka data waktu pengujian pada kolom *next_test* yang terdapat pada tabel *server_testcase* harus diperbarui. Data waktu baru yang dimasukkan adalah waktu pengujian terakhir pada tabel ditambah dengan nilai *interval* (dalam satuan menit). Hal ini dikarenakan pada saat pengujian selanjutnya untuk pemasangan kasus uji akan dilakukan, parameter yang digunakan untuk pemilihan pemasangan kasus uji salah satunya adalah waktu pada kolom *next_text.* Jika kolom ini tidak diperbarui, maka pada pengujian selanjutnya dimana pemasangan kasus uji ini seharusnya terpilih untuk dieksekusi, pemasangan pengujian kasus uji tidak akan terpilih.

Fungsi *updateNextTest* berguna untuk memperbarui data waktu pengujian selanjutnya pada kasus uji. Parameter yang diperlukan untuk memanggil fungsi ini yaitu *primary key* dari tabel *server_testcase*.

Query yang digunakan untuk melakukan update pada kolom *next_text* terdapat pada kode sumber 4.8.

```
update st1
set st1.NEXT_TEST= DATEADD(minute,st2.interval ,st2.next_test)
FROM [dbo].[SERVER_TESTCASE]st1
inner join [dbo].[SERVER_TESTCASE] st2
on st1.id_server_testcase = st2.id_server_testcase
where
st1.ID_SERVER_TESTCASE=”testcase_server_id”
```

Kode Sumber 4.8 *Query* untuk memperbarui pengujian terakhir

4.3.2. Implementasi Pengiriman Perintah Pengujian

Proses pengiriman perintah pengujian ke server uji dieksekusi oleh *Scheduler engine* pada *server* pusat. Proses ini adalah implementasi dari kasus penggunaan SP01.

Proses *Scheduler engine* membuat penjadwalan yang memiliki fungsi melakukan pengiriman perintah pengujian pada *Server* Uji setiap menit secara berulang-ulang.

Proses Dimulai dari sebuah *class* yang bernama *scheduler* pada perangkat lunak *Scheduler Engine*. Di dalam *class* ini,

terdapat fungsi *startScheduler* yang digunakan untuk menginisiasi *scheduler* pada perangkat lunak *Scheduler Engine.*

Fungsi *startScheduler* diinisiasi dengan mengakses *endpoint* dengan struktur seperti pada gambar 4.7.

```
{Alamat Server Pusat} /web_service_monitoring/ webapi/ scheduler/ start
```

Gambar 4.7 Struktur *Endpoint* untuk Inisiasi *Scheduler* Pada *Scheduler Engine*

Endpoint pada gambar diatas diakses dengan menggunakan protocol HTTP GET, sesuai dengan deklarasi *endpoint* fungsi *startScheduler* milik *Request Engine* pada *server* pusat yang ditunjukkan pada kode sumber 4.9.

```
@Path ("/scheduler")
public class scheduler {
        @GET
        @Path("/start")
        @Produces(MediaType.TEXT_PLAIN)
        public String startScheduler() {}
```

Kode Sumber 4.9 Deklarasi *Endpoint* untuk menyalakan *Scheduler*

Kode semu fungsi *startScheduler* terdapat pada gambar 4.8.

1	startScheduler()
2	new scheduler_executor()
3	scheduler_executor.main()

Gambar 4.8 Kode Semu Fungsi *StartScheduler*

Pada baris nomor 2 dan 3, terdapat instansiasi *class scheduler_excecutor* dan pemanggilan fungsi *main* pada *class* tersebut. Fungsi *main* pada *class scheduler_executor* memiliki fungsi untuk menjalankan *scheduler* dengan menggunakan *library timer* yang tersedia pada Bahasa pemrograman Java.

Kode semu fungsi *main* dapat dilihat pada gambar 4.9.

1	main()
2	scheduler_function = new scheduler_function() //pendefinisian Class tugas pengiriman
3	Timer.Schedule(scheduler_function(), nowTime, interval) //set *scheduler*

Gambar 4.9 Kode Semu Fungsi *main*

Pada baris nomor 2 dan 3, terdapat instansiasi *class scheduler_function* dan pemanggilan fungsi *scheduler* dengan hasil instansiasinya. *Class scheduler_function* adalah perpanjangan *class* dari *class timerTask* pada Bahasa pemrograman Java. *Class* ini dapat digunakan sebagai fungsi yang dieksekusi secara berkala. *nowTime* dan *interval* pada baris nomor 3 masing-masing adalah waktu awal inisiasi pengiriman perintah pengujian dan selang waktu antar inisiasi pengiriman perintah pengujian.

Pada *class scheduler_function* terdapat dua bagian, yaitu fungsi *run* (selalu dipanggil otomatis secara berkala jika dijadikan *scheduler*) untuk menginisiasi pengiriman berdasarkan daftar server yang harus diberi perintah pengujian, dan *inner class runtrhead*, yang dapat mengirimkan perintah pengujian dalam *thread* yang terpisah-pisah.

Pada fungsi *run*, hanya terdapat satu perintah, yaitu pemanggilan fungsi *start_thread*. Fungsi *start_thread* berfungsi untuk mengumpulkan daftar *server* uji yang akan diuji, kemudian membuat pengiriman perintah pengujian pada masing-masing *server uji*. Kode semu fungsi *start_thread* terdapat pada gambar 4.10.

1	Start_thread()
2	data=getservers() // Mengumpulkan daftar server yang akan diberi perintah eksekusi pengujian
3	For data as d:
4	New runthread(i.address)
5	Runthread.start()

Gambar 4.10 Kode Semu Fungsi Start_thread

Pada baris nomor 2, terdapat bagian yang berfungsi untuk mengambil daftar server uji yang akan diberi perintah pengujian dengan *query. Query* yang digunakan terdapat pada kode sumber 4.10.

```
1   SELECT * FROM [dbo].[server] s WHERE s.active=1
```

Kode Sumber 4.10 Query untuk Mendapatkan Daftar *Server Uji*

Query tersebut mengambil data *server* uji yang akan diberi perintah. Kriteria *server* uji yang akan diberi perintah yaitu *server* uji yang memiliki status aktif =1. Kolom *address* dari hasil *query* akan digunakan pada instansiasi *inner class runthread.*

Pada baris nomor 4 dan 5, terdapat instansiasi *inner class runthread* dengan parameter dan aktivasi *thread* pada hasil instansiasi. Pada *inner class runthread,* terdapat fungsi *run* yang dijalankan secara otomatis ketika terdapat aktivasi *thread.* Kode semu fungsi *run* pada *inner class runthread* terdapat pada gambar 4.11.

```
1   Class runthread
2     Function run (address)
3           build(Request)(i)//membuat request perintah
4           send(Request)(i)//mengirim request perintah
```

Gambar 4.11 Kode Semu Eksekusi Pengiriman Perintah Pengujian

Pada fungsi *buidRequest* dan *sendRequest,* akan dibentuk sebuah GET *request* yang akan dikirim pada *Request Engine* yang terdapat pada *server* uji. URL *endpoint* pada request yang terdapat pada kedua fungsi ini terdapat pada gambar 4.12.

```
{Alamat server uji} / web_service_monitoring/
webapi/entrypoint?time={timestamp} &server={address}
```

Gambar 4.12 URL Pengiriman Perintah Pengujian

Pengiriman *request* GET dengan URL seperti pada gambar 4.12 akan menginisiasi proses eksekusi pengujian kasus uji, sesuai

dengan proses yang dibahas pada bagian implementasi tahap pengiriman *request*.

4.4. Implementasi Antarmuka Perangkat Lunak

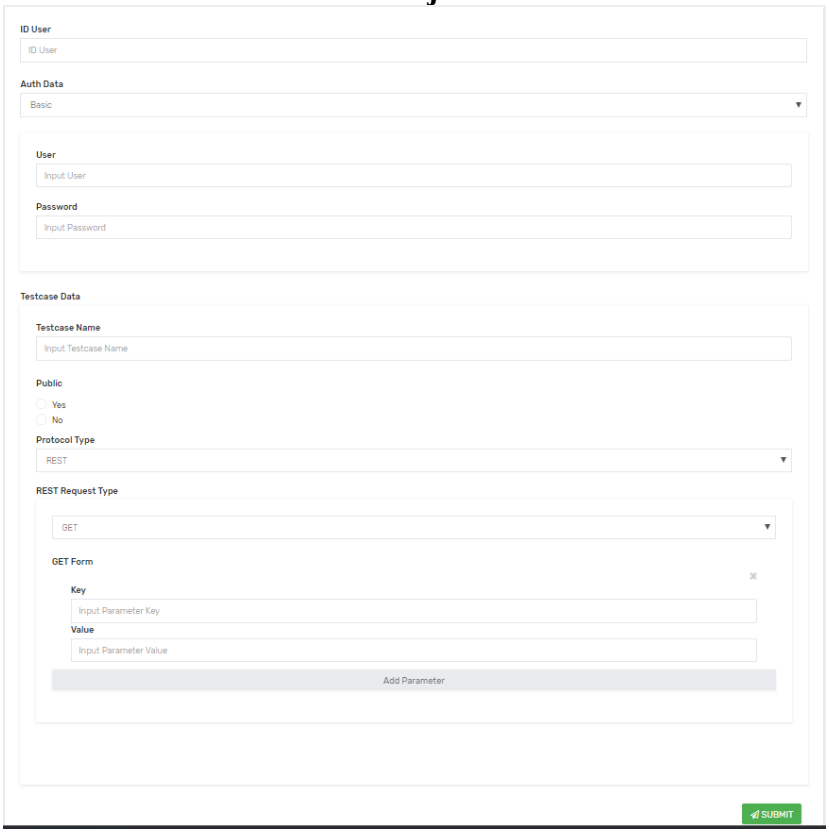
Pada subbab ini akan dibahas mengenai implementasi tampilan antarmuka perangkat lunak pada tugas akhir ini. Desain antarmuka perangkat lunak pada subbab ini didasari pada rancangan-rancangan antarmuka perangkat lunak yang terdapat pada bagian perancangan antarmuka perangkat lunak. Pada sistem ini, hanya terdapat satu perangkat lunak yang memiliki tampilan antarmuka, yaitu Aplikasi *Dashboard.* Implementasi antarmuka perangkat lunak pada aplikasi *Dashboard* dibuat dengan bahasa pemrograman PHP berbasis kerangka kerja *codeIgniter.*

Implementasi tampilan antarmuka yang terdapat pada aplikasi *Dashboard* akan ditampilkan pada bagian-bagian berikut ini.

4.4.1. Implementasi Antarmuka Halaman Hasil Pengujian pada *Server* Uji

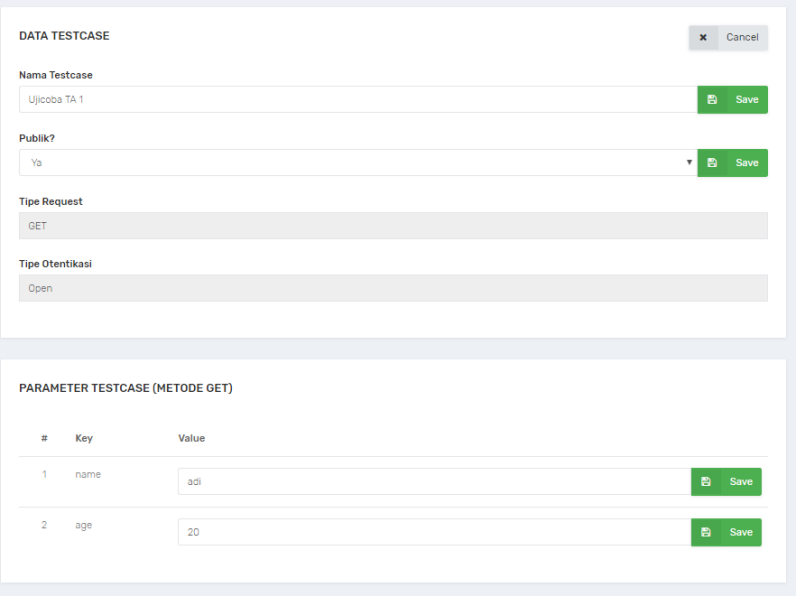
Gambar 4.13 Implementasi Antarmuka Halaman Hasil Pengujian Pada *Server* Uji

4.4.2. Implementasi Antarmuka Halaman Penambahan Kasus Uji



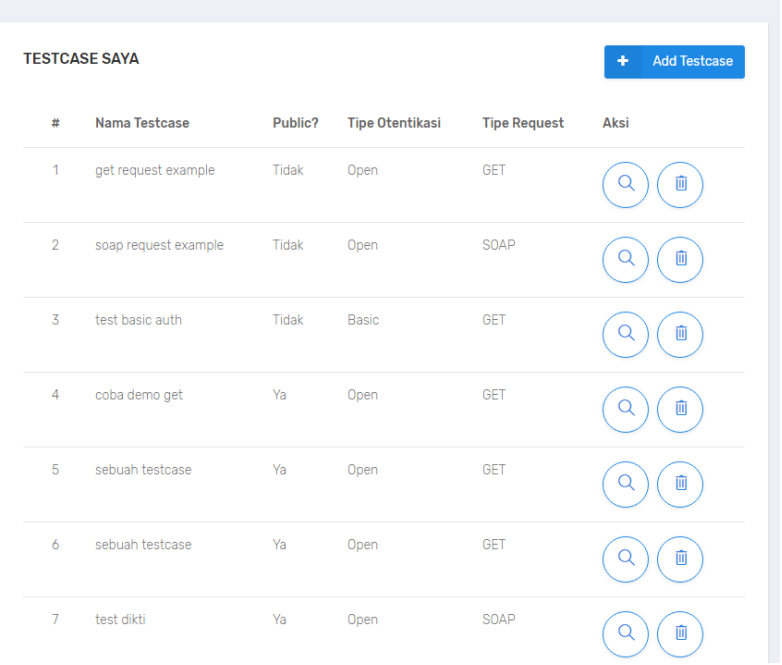
Gambar 4.14 Implementasi Antarmuka Halaman Penambahan Kasus Uji

4.4.3. Implementasi Antarmuka Halaman Ubah Detail Kasus Uji



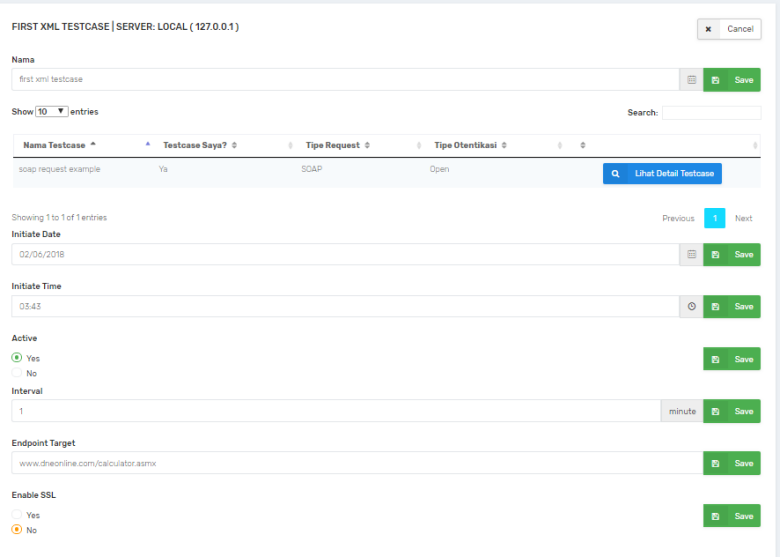
Gambar 4.15 Implementasi Antarmuka Halaman Ubah Detail Kasus Uji

4.4.4. Implementasi Antarmuka Halaman Pemasangan Kasus Uji Pada *Server* Uji



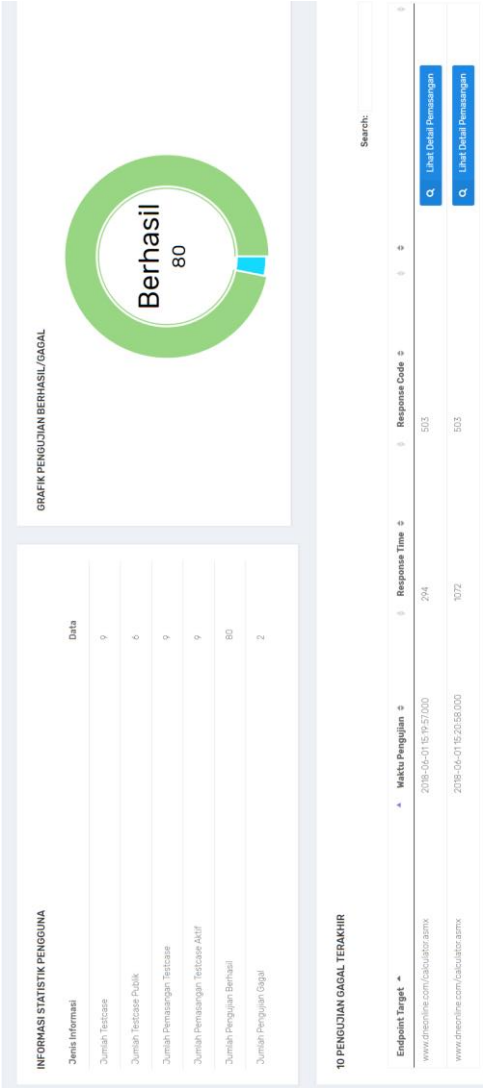
Gambar 4.16 Implementasi Antarmuka Halaman Pemasangan Kasus Uji Pada *Server* Uji

4.4.5. Implementasi Antarmuka Halaman Mengubah Detail Pemasangan Kasus Uji pada Server Uji



Gambar 4.17 Implementasi Antarmuka Halaman Mengubah Detail Pemasangan Kasus Uji pada *Server* Uji

4.4.6. Implementasi Antarmuka Halaman *Dashboard*



Gambar 4.18 Implementasi Antarmuka Halaman *Dashboard*

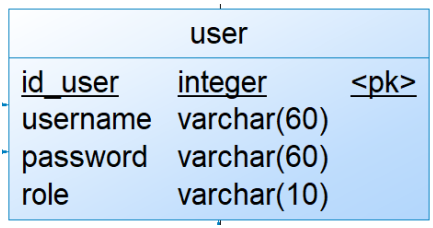
4.5. Implementasi Basis Data

Tabel-tabel implementasi basis data yang terdapat pada pada subbab-subbab di bawah ini adalah implementasi dari tabel-tabel yang terdapat pada bagian perancangan basis data pada bab sebelumnya.

4.5.1. Implementasi Tabel *User*

```
CREATE TABLE [dbo].[USER](
	[ID_USER] [int] IDENTITY(1,1) NOT NULL,
	[USERNAME] [varchar](60) NULL,
	[PASSWORD] [varchar](60) NULL,
	[ROLE] [varchar](10) NULL,
 CONSTRAINT [PK_USER] PRIMARY KEY CLUSTERED
(
	[ID_USER] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Kode Sumber 4.11 Tabel *User*



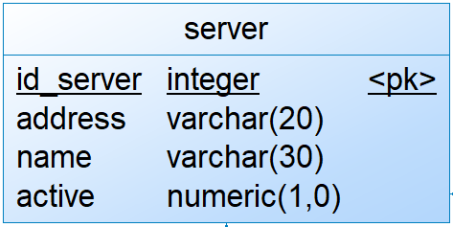
Gambar 4.19 Tabel *User*

4.5.2. Tabel *Server*

```
CREATE TABLE [dbo].[SERVER](
	[ID_SERVER] [int] IDENTITY(1,1) NOT NULL,
	[ADDRESS] [varchar](20) NULL,
	[NAME] [varchar](30) NULL,
```

```
       [ACTIVE] [numeric](1, 0) NULL,
 CONSTRAINT [PK_SERVER] PRIMARY KEY CLUSTERED
(
       [ID_SERVER] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Kode Sumber 4.12 Tabel *Server*



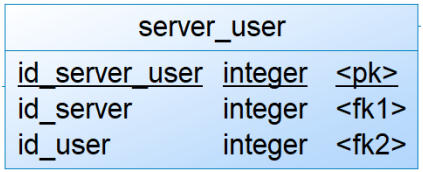
Gambar 4.20 Tabel *Server*

4.5.3. Tabel Server User

```
CREATE TABLE [dbo].[SERVER_USER](
       [ID_SERVER_USER] [int] IDENTITY(1,1) NOT NULL,
       [ID_SERVER] [int] NULL,
       [ID_USER] [int] NULL,
       [ACTIVE] [int] NULL,
 CONSTRAINT [PK_SERVER_USER] PRIMARY KEY CLUSTERED
(
       [ID_SERVER_USER] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[SERVER_USER] ADD  CONSTRAINT
[DF_SERVER_USER_ACTIVE]  DEFAULT ((0)) FOR [ACTIVE]
GO
```

```
ALTER TABLE [dbo].[SERVER_USER]  WITH NOCHECK ADD
CONSTRAINT [FK_SERVER_U_REFERENCE_SERVER] FOREIGN
KEY([ID_SERVER])
REFERENCES [dbo].[SERVER] ([ID_SERVER])
GO

ALTER TABLE [dbo].[SERVER_USER] NOCHECK CONSTRAINT
[FK_SERVER_U_REFERENCE_SERVER]
GO

ALTER TABLE [dbo].[SERVER_USER]  WITH NOCHECK ADD
CONSTRAINT [FK_SERVER_U_REFERENCE_USER] FOREIGN
KEY([ID_USER])
REFERENCES [dbo].[USER] ([ID_USER])
GO

ALTER TABLE [dbo].[SERVER_USER] NOCHECK CONSTRAINT
[FK_SERVER_U_REFERENCE_USER]
GO
```

Kode Sumber 4.13 Tabel *Server_User*



Gambar 4.21 Tabel *Server_User*

4.5.4. Tabel *Testcase*

```
CREATE TABLE [dbo].[TESTCASE](
       [ID_TESTCASE] [int] IDENTITY(1,1) NOT NULL,
       [ID_REQUEST_TYPE] [int] NULL,
       [ID_USER] [int] NULL,
       [TESTCASE_NAME] [varchar](100) NULL,
       [IS_PUBLIC] [numeric](1, 0) NULL,
       [SOFT_DELETE] [int] NULL,
 CONSTRAINT [PK_TESTCASE] PRIMARY KEY CLUSTERED
(
```

```
       [ID_TESTCASE] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[TESTCASE] ADD  CONSTRAINT
[DF_TESTCASE_SOFT_DELETE]  DEFAULT ((0)) FOR
[SOFT_DELETE]
GO

ALTER TABLE [dbo].[TESTCASE]  WITH NOCHECK ADD
CONSTRAINT [FK_TESTCASE_REFERENCE_REQUEST_2] FOREIGN
KEY([ID_REQUEST_TYPE])
REFERENCES [dbo].[REQUEST_TYPE] ([ID_REQUEST_TYPE])
GO

ALTER TABLE [dbo].[TESTCASE] NOCHECK CONSTRAINT
[FK_TESTCASE_REFERENCE_REQUEST_2]
GO

ALTER TABLE [dbo].[TESTCASE]  WITH NOCHECK ADD
CONSTRAINT [FK_TESTCASE_REFERENCE_USER] FOREIGN
KEY([ID_USER])
REFERENCES [dbo].[USER] ([ID_USER])
GO

ALTER TABLE [dbo].[TESTCASE] NOCHECK CONSTRAINT
[FK_TESTCASE_REFERENCE_USER]
GO
```

Kode Sumber 4.14 Tabel *Testcase*

testcase		
id_testcase	integer	<pk>
id_request_type	integer	<fk1>
id_user	integer	<fk2>
testcase_name	varchar(100)	
is_public	numeric(1,0)	

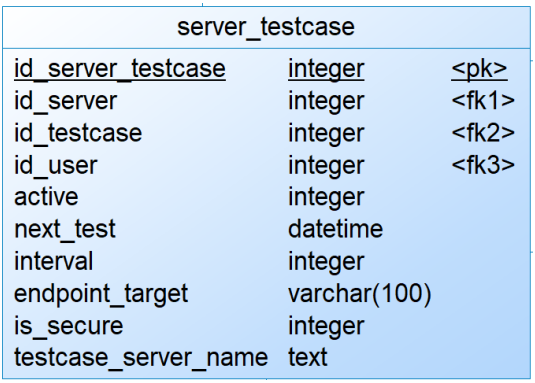
Gambar 4.22 Tabel *Testcase*

4.5.5. Tabel *Server_Testcase*

```
CREATE TABLE [dbo].[SERVER_TESTCASE](
	[ID_SERVER_TESTCASE] [int] IDENTITY(1,1) NOT
NULL,
	[ID_SERVER] [int] NULL,
	[ID_TESTCASE] [int] NULL,
	[SERVER_TESTCASE_NAME] [text] NULL,
	[ACTIVE] [int] NULL,
	[NEXT_TEST] [datetime] NULL,
	[INTERVAL] [int] NULL,
	[ENDPOINT_TARGET] [varchar](250) NULL,
	[IS_SECURE] [int] NULL,
	[ID_USER] [int] NULL,
	[SOFT_DELETE] [int] NULL,
 CONSTRAINT [PK_SERVER_TESTCASE] PRIMARY KEY
CLUSTERED
(
	[ID_SERVER_TESTCASE] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[SERVER_TESTCASE] ADD  CONSTRAINT
[DF_SERVER_TESTCASE_SOFT_DELETE]  DEFAULT ((0)) FOR
[SOFT_DELETE]
GO
```

```
ALTER TABLE [dbo].[SERVER_TESTCASE]  WITH NOCHECK ADD
CONSTRAINT [FK_SERVER_T_REFERENCE_SERVER] FOREIGN
KEY([ID_SERVER])
REFERENCES [dbo].[SERVER] ([ID_SERVER])
GO

ALTER TABLE [dbo].[SERVER_TESTCASE] NOCHECK
CONSTRAINT [FK_SERVER_T_REFERENCE_SERVER]
GO

ALTER TABLE [dbo].[SERVER_TESTCASE]  WITH NOCHECK ADD
CONSTRAINT [FK_SERVER_T_REFERENCE_TESTCASE] FOREIGN
KEY([ID_TESTCASE])
REFERENCES [dbo].[TESTCASE] ([ID_TESTCASE])
GO

ALTER TABLE [dbo].[SERVER_TESTCASE] NOCHECK
CONSTRAINT [FK_SERVER_T_REFERENCE_TESTCASE]
GO
```

Kode Sumber 4.15 Tabel *Server_Testcase*



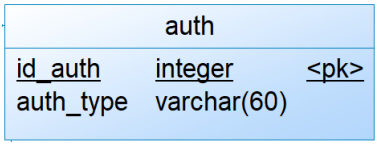
Gambar 4.23 Tabel *Server_Testcase*

4.5.6. Tabel *Auth*

```
CREATE TABLE [dbo].[AUTH](
       [ID_AUTH] [int] IDENTITY(1,1) NOT NULL,
```

```
	[AUTH_TYPE] [varchar](60) NULL,
 CONSTRAINT [PK_AUTH] PRIMARY KEY CLUSTERED
(
	[ID_AUTH] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Kode Sumber 4.16 Tabel *Auth*



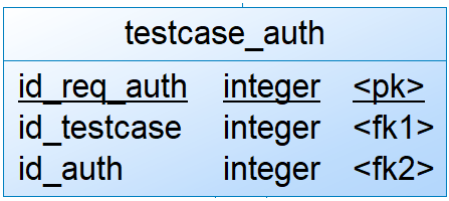
Gambar 4.24 Tabel *Auth*

4.5.7. Tabel *Testcase_Auth*

```
CREATE TABLE [dbo].[TESTCASE_AUTH](
	[ID_REQ_AUTH] [int] IDENTITY(1,1) NOT NULL,
	[ID_TESTCASE] [int] NULL,
	[ID_AUTH] [int] NULL,
 CONSTRAINT [PK_TESTCASE_AUTH] PRIMARY KEY CLUSTERED
(
	[ID_REQ_AUTH] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[TESTCASE_AUTH]  WITH NOCHECK ADD
CONSTRAINT [FK_TESTCASE_REFERENCE_AUTH] FOREIGN
KEY([ID_AUTH])
REFERENCES [dbo].[AUTH] ([ID_AUTH])
GO
```

```
ALTER TABLE [dbo].[TESTCASE_AUTH] NOCHECK CONSTRAINT
[FK_TESTCASE_REFERENCE_AUTH]
GO

ALTER TABLE [dbo].[TESTCASE_AUTH]  WITH NOCHECK ADD
CONSTRAINT [FK_TESTCASE_REFERENCE_TESTCASE] FOREIGN
KEY([ID_TESTCASE])
REFERENCES [dbo].[TESTCASE] ([ID_TESTCASE])
GO

ALTER TABLE [dbo].[TESTCASE_AUTH] NOCHECK CONSTRAINT
[FK_TESTCASE_REFERENCE_TESTCASE]
GO
```

Kode Sumber 4.17 Tabel *Testcase_Auth*



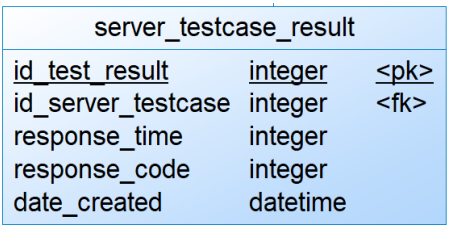
Gambar 4.25 Tabel *Testcase_Auth*

4.5.8. Tabel *Server_Testcase_Result*

```
CREATE TABLE [dbo].[SERVER_TESTCASE_RESULT](
	[ID_TEST_RESULT] [int] IDENTITY(1,1) NOT NULL,
	[ID_SERVER_TESTCASE] [int] NULL,
	[RESPONSE_TIME] [int] NULL,
	[RESPONSE_CODE] [int] NULL,
	[DATE_CREATED] [datetime] NULL,
 CONSTRAINT [PK_SERVER_TESTCASE_RESULT] PRIMARY KEY
CLUSTERED
(
	[ID_TEST_RESULT] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[SERVER_TESTCASE_RESULT]  WITH
NOCHECK ADD  CONSTRAINT
[FK_SERVER_T_REFERENCE_SERVER_T] FOREIGN
KEY([ID_SERVER_TESTCASE])
REFERENCES [dbo].[SERVER_TESTCASE]
([ID_SERVER_TESTCASE])
GO

ALTER TABLE [dbo].[SERVER_TESTCASE_RESULT] NOCHECK
CONSTRAINT [FK_SERVER_T_REFERENCE_SERVER_T]
GO
```

Kode Sumber 4.18 Tabel *Server_Testcase_Result*



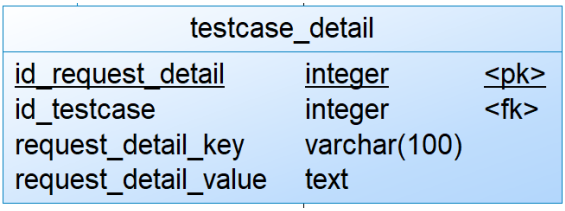
Gambar 4.26 Tabel *Server_Testcase_Result*

4.5.9. Tabel *Testcase_Detail*

```
CREATE TABLE [dbo].[TESTCASE_DETAIL](
	[ID_REQUEST_DETAIL] [int] IDENTITY(1,1) NOT
NULL,
	[ID_TESTCASE] [int] NULL,
	[REQUEST_DETAIL_KEY] [varchar](100) NULL,
	[REQUEST_DETAIL_VALUE] [text] NULL,
 CONSTRAINT [PK_TESTCASE_DETAIL] PRIMARY KEY
CLUSTERED
(
	[ID_REQUEST_DETAIL] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[TESTCASE_DETAIL]  WITH NOCHECK ADD
CONSTRAINT [FK_TESTCASE_REFERENCE_TESTCASE2] FOREIGN
KEY([ID_TESTCASE])
REFERENCES [dbo].[TESTCASE] ([ID_TESTCASE])
GO

ALTER TABLE [dbo].[TESTCASE_DETAIL] NOCHECK
CONSTRAINT [FK_TESTCASE_REFERENCE_TESTCASE2]
GO
```

Kode Sumber 4.19 Tabel *Testcase_Detail*



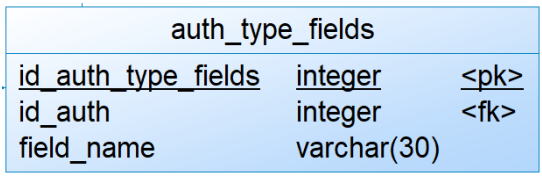
Gambar 4.27 Tabel *Testcase_Detail*

4.5.10. Tabel *Auth_Type_Fields*

```
CREATE TABLE [dbo].[AUTH_TYPE_FIELDS](
	[ID_AUTH_TYPE_FIELDS] [int] IDENTITY(1,1) NOT
NULL,
	[ID_AUTH] [int] NULL,
	[FIELD_NAME] [varchar](30) NULL,
 CONSTRAINT [PK_AUTH_TYPE_FIELDS] PRIMARY KEY
CLUSTERED
(
	[ID_AUTH_TYPE_FIELDS] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[AUTH_TYPE_FIELDS]  WITH NOCHECK
ADD  CONSTRAINT [FK_AUTH_TYP_REFERENCE_AUTH] FOREIGN
KEY([ID_AUTH])
REFERENCES [dbo].[AUTH] ([ID_AUTH])
GO

ALTER TABLE [dbo].[AUTH_TYPE_FIELDS] NOCHECK
CONSTRAINT [FK_AUTH_TYP_REFERENCE_AUTH]
GO
```

Kode Sumber 4.20 Tabel *Auth_Type_Fields*



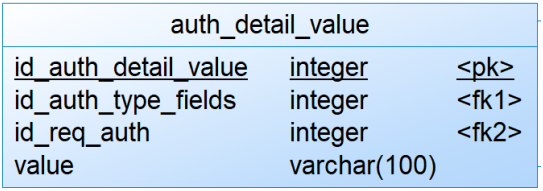
Gambar 4.28 Tabel *Auth_Type_Fields*

4.5.11. Tabel *Auth_Detail_Value*

```
CREATE TABLE [dbo].[AUTH_DETAIL_VALUE](
       [ID_AUTH_DETAIL_VALUE] [int] IDENTITY(1,1) NOT
NULL,
       [ID_AUTH_TYPE_FIELDS] [int] NULL,
       [ID_REQ_AUTH] [int] NULL,
       [VALUE] [varchar](100) NULL,
 CONSTRAINT [PK_AUTH_DETAIL_VALUE] PRIMARY KEY
CLUSTERED
(
       [ID_AUTH_DETAIL_VALUE] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AUTH_DETAIL_VALUE]  WITH NOCHECK
ADD  CONSTRAINT [FK_AUTH_DET_REFERENCE_AUTH_TYP]
FOREIGN KEY([ID_AUTH_TYPE_FIELDS])
```

```
REFERENCES [dbo].[AUTH_TYPE_FIELDS]
([ID_AUTH_TYPE_FIELDS])
GO

ALTER TABLE [dbo].[AUTH_DETAIL_VALUE] NOCHECK
CONSTRAINT [FK_AUTH_DET_REFERENCE_AUTH_TYP]
GO

ALTER TABLE [dbo].[AUTH_DETAIL_VALUE]  WITH NOCHECK
ADD  CONSTRAINT [FK_AUTH_DET_REFERENCE_TESTCASE]
FOREIGN KEY([ID_REQ_AUTH])
REFERENCES [dbo].[TESTCASE_AUTH] ([ID_REQ_AUTH])
GO

ALTER TABLE [dbo].[AUTH_DETAIL_VALUE] NOCHECK
CONSTRAINT [FK_AUTH_DET_REFERENCE_TESTCASE]
GO
```

Kode Sumber 4.21 Tabel *Auth_Detail_Value*



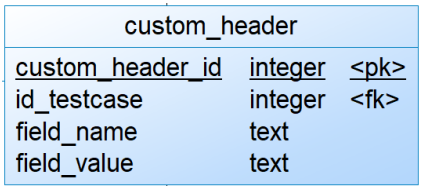
Gambar 4.29 Tabel *Auth_Detail_Value*

4.5.12. Tabel *Custom_Header*

```
CREATE TABLE [dbo].[CUSTOM_HEADER](
	[ID_SOAP_AUTH] [int] NOT NULL,
	[USERNAME] [varchar](100) NULL,
	[TOKEN] [varchar](100) NULL,
	[ID_TESTCASE] [int] NULL,
 CONSTRAINT [PK_SOAP_AUTH] PRIMARY KEY CLUSTERED
(
	[ID_SOAP_AUTH] ASC
```

```
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[CUSTOM_HEADER]  WITH CHECK ADD
CONSTRAINT [FK_SOAP_AUTH_TESTCASE] FOREIGN
KEY([ID_TESTCASE])
REFERENCES [dbo].[TESTCASE] ([ID_TESTCASE])
GO

ALTER TABLE [dbo].[CUSTOM_HEADER] CHECK CONSTRAINT
[FK_SOAP_AUTH_TESTCASE]
GO
```

Kode Sumber 4.22 Tabel *Custom_Header*



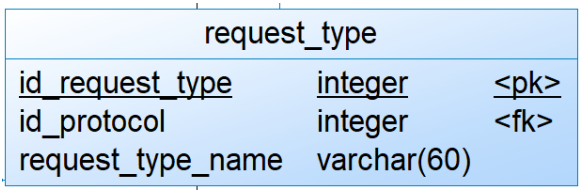
Gambar 4.30 Tabel *Custom_Header*

4.5.13. Tabel *Request_Type*

```
CREATE TABLE [dbo].[REQUEST_TYPE](
	[ID_REQUEST_TYPE] [int] IDENTITY(1,1) NOT
NULL,
	[ID_PROTOCOL] [int] NULL,
	[REQUEST_TYPE_NAME] [varchar](60) NULL,
 CONSTRAINT [PK_REQUEST_TYPE] PRIMARY KEY CLUSTERED
(
	[ID_REQUEST_TYPE] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

```
ALTER TABLE [dbo].[REQUEST_TYPE]  WITH NOCHECK ADD
CONSTRAINT [FK_REQUEST__REFERENCE_PROTOCOL] FOREIGN
KEY([ID_PROTOCOL])
REFERENCES [dbo].[PROTOCOL] ([ID_PROTOCOL])
GO

ALTER TABLE [dbo].[REQUEST_TYPE] NOCHECK CONSTRAINT
[FK_REQUEST__REFERENCE_PROTOCOL]
GO
```

Kode Sumber 4.23 Tabel *Request_Type*

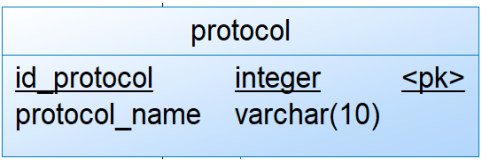


Gambar 4.31 Tabel *Request_Type*

4.5.14. Tabel *Protocol*

```
CREATE TABLE [dbo].[PROTOCOL](
       [ID_PROTOCOL] [int] IDENTITY(1,1) NOT NULL,
       [PROTOCOL_NAME] [varchar](10) NULL,
 CONSTRAINT [PK_PROTOCOL] PRIMARY KEY CLUSTERED
(
       [ID_PROTOCOL] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Kode Sumber 4.24 Tabel *Protocol*



Gambar 4.32 Tabel *Protocol*

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas mengenai uji coba dan evaluasi terhadap perangkat lunak yang telah dikembangkan dari implementasi Aplikasi Pemantauan *Web Service* berbasis protokol REST dan SOAP. Pengujian yang dilakukan pada system yaitu pengujian fungsionalitas dengan skenario-skenario yang dibuat dengan mengacu pada kasus penggunaan.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

a. *Server* Pusat
   Prosesor : Intel® Core i7 7700HQ (4 Cores @ 2.80GHz)
   RAM : 16384 MB
   Jenis *Device* : *Notebook*
   Sistem Operasi : Microsoft Windows 10 SL 64 bit
b. *Server* Uji
   Prosesor : Intel® Core i7 4500U (2 Cores @ 1.80GHz)
   RAM : 4096 MB
   Jenis *Device* : *Notebook*
   Sistem Operasi : Microsoft Windows 10 Pro 64 bit
c. Wifi : *MiFi* Huawei E5577

5.2. Skenario Uji Coba Fungsional

Pada subbab ini akan dijelaskan meneganai skenario uji coba yang telah dilakukan. Pada beberapa skenario pengujian, terdapat lebih dari satu kasus pengujian yang menjadi referensi kasus penggunaan. Hal tersebut dikarenakan beberapa kasus penggunaan saling berkesinambungan dan perlu dilakukan secara

berurutan. Terdapat beberapa skenario uji coba yang telah dilakukan, diantaranya yaitu sebagai berikut.

1. Skenario pengujian 1: Membuat kasus uji dengan metode HTTP GET.
2. Skenario pengujian 2: Membuat kasus uji dengan metode HTTP POST.
3. Skenario pengujian 3: Membuat kasus uji dengan protokol SOAP.
4. Skenario pengujian 4: Membuat kasus uji dengan metode HTTP GET dan dengan metode otentikasi Basic.
5. Skenario pengujian 5: Membuat kasus uji dengan metode HTTP GET dan dengan metode otentikasi Digest.
6. Skenario pengujian 6: Melihat detail kasus uji yang Telah Dibuat.
7. Skenario pengujian 7: Mengubah detail kasus uji yang Telah Dibuat.
8. Skenario pengujian 8: Menonaktifkan kasus uji
9. Skenario pengujian 9: Memasang kasus uji pada *server* uji sebagai pengujian.
10. Skenario pengujian 10: Mengubah detail pemasangan kasus uji.
11. Skenario pengujian 11: Menonaktifkan pemasangan pengujian.
12. Memberi Akses pengguna ke *server* uji.
13. Melihat hasil pengujian pemasangan kasus uji pada server uji.

5.2.1. Skenario Pengujian 1: Membuat Kasus Uji dengan Metode HTTP GET

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01.

Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.1.

Tabel 5.1 Tabel Skenario Pengujian Membuat Kasus Uji dengan Metode HTTP GET

Referensi Kasus Penggunaan	DU01
Nama	Pengujian membuat kasus uji dengan metode HTTP GET
Tujuan Pengujian	Membuat kasus uji dengan metode GET
Skenario 1	**Pengguna membuat kasus uji dengan metode HTTP GET**
Kondisi Awal	Pengguna berada pada antarmuka penambahan kasus uji
Data Uji	HTTP GET *request* Parameter: Testcase name: ujicoba TA 1 name=adi age=20
Langkah Pengujian	-Pengguna membuka menu untuk menambahkan kasus uji -Pengguna mengisi nama kasus uji -Pengguna memilih tipe otentikasi “open” -Pengguna memilih tipe protocol REST -Pengguna memilih tipe *request* GET -Pengguna memasukkan parameter yang ada -Pengguna menyimpan kasus uji.
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan akan muncul pada halaman daftar kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan sudah tersimpan

5.2.2. Skenario Pengujian 2: Membuat Kasus Uji dengan Metode HTTP POST

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.2.

Tabel 5.2 Skenario Pengujian Membuat Kasus Uji dengan Metode HTTP POST

Referensi Kasus Penggunaan	DU01
Nama	Pengujian membuat kasus uji
Tujuan Pengujian	Membuat kasus uji dengan metode HTTP POST
Skenario 1	**Pengguna membuat kasus uji dengan metode HTTP POST**
Kondisi Awal	Pengguna berada pada antarmuka penambahan kasus uji
Data Uji	HTTP POST *request* Parameter: Testcase name: postman POST request Request parameter: name=adi age=20
Langkah Pengujian	-Pengguna membuka menu untuk menambahkan kasus uji -Pengguna mengisi nama kasus uji -Pengguna memilih tipe otentikasi “open” -Pengguna memilih tipe protocol REST -Pengguna memilih tipe *request* POST -Pengguna memasukkan parameter yang ada -Pengguna menyimpan kasus uji.
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan akan muncul pada halaman daftar kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji

	-data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan telah tersimpan

5.2.3. Skenario Pengujian 3: Membuat Kasus Uji dengan Protokol SOAP

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.3.

Tabel 5.3 Skenario Pengujian Membuat Kasus Uji dengan Metode SOAP

Referensi Kasus Penggunaan	DU01
Nama	Pengujian membuat kasus uji dengan protokol SOAP
Tujuan Pengujian	Membuat kasus uji dengan protokol SOAP
Skenario 1	**Pengguna membuat kasus uji dengan protokol SOAP**
Kondisi Awal	Pengguna berada pada halaman antarmuka penambahan kasus uji
Data Uji	SOAP *request* Parameter: -Testcase name: ujicoba SOAP - SOAP Action = http://tempuri.org/Add -XML text = <?xml version="1.0" encoding="utf-8"?> <soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <Add xmlns="http://tempuri.org/"> <intA>3</intA>

	<intB>4</intB> </Add> </soap:Body> </soap:Envelope>
Langkah Pengujian	-Pengguna membuka menu untuk menambahkan kasus uji -Pengguna mengisi nama kasus uji -Pengguna memilih tipe otentikasi “open” -Pengguna memilih tipe protocol SOAP -Pengguna memasukkan parameter yang ada -Pengguna menyimpan kasus uji.
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan akan muncul pada halaman daftar kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan sudah tersimpan

5.2.4. Skenario Pengujian 4: Membuat Kasus Uji dengan Metode HTTP GET dan Metode Otentikasi Basic

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.4.

Tabel 5.4 Skenario Pengujian Membuat Kasus Uji dengan Metode HTTP GET dengan Metode Otentikasi Basic

Referensi Kasus Penggunaan	DU01
Nama	Pengujian membuat kasus uji dengan metode HTTP GET dan metode otentikasi Basic
Tujuan Pengujian	Membuat kasus uji dengan metode GET dan metode otentikasi Basic

Skenario 1	**Pengguna membuat kasus uji dengan metode HTTP GET dan metode otentikasi Basic**
Kondisi Awal	Pengguna berada pada antarmuka penambahan kasus uji
Data Uji	HTTP GET *request* Parameter: Testcase name: ujicoba Basic Data otentikasi basic: Username=”postman” Passsword=”password”
Langkah Pengujian	-Pengguna membuka menu untuk menambahkan kasus uji -Pengguna mengisi nama kasus uji -Pengguna memilih tipe otentikasi “Basic” -Pengguna memasukkan *username* dan *password* untuk otentikasi -Pengguna memilih tipe *protocol* REST -Pengguna memilih tipe *request* GET -Pengguna menyimpan kasus uji.
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan akan muncul pada halaman daftar kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan sudah tersimpan

5.2.5. Skenario Pengujian 5: Membuat Kasus Uji dengan Metode HTTP GET dan dengan Metode Otentikasi *Digest*

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01.

Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.5.

Tabel 5.5 Skenario Pengujian Membuat Kasus Uji dengan Metode HTTP GET dan dengan Metode Otentikasi *Digest*

Referensi Kasus Penggunaan	DU01
Nama	Pengujian membuat kasus uji dengan metode HTTP GET
Tujuan Pengujian	Membuat kasus uji dengan metode GET dan metode otentikasi D*igest*
Skenario 1	**Pengguna membuat kasus uji dengan metode HTTP GET dan metode otentikasi D*igest***
Kondisi Awal	Pengguna berada pada antarmuka penambahan kasus uji
Data Uji	HTTP GET *request* Parameter: Testcase name: ujicoba digest postman Data otentikasi Digest: username="postman", realm="Users", nonce="LztT2EMJfBeNfZgJW7oLT6HWG4Y9wxdc", uri="/digest-auth", algorithm="MD5-sess", qop=auth, nc=00000001, cnonce="asdasdasdasd", response="debeac466c7414c830d28bf20f626431"
Langkah Pengujian	Pengguna membuka menu untuk menambahkan kasus uji -Pengguna mengisi nama kasus uji -Pengguna memilih tipe otentikasi “Digest” -Pengguna memasukkan data-data untuk otentikasi Digest -Pengguna memilih tipe *protocol* REST -Pengguna memilih tipe *request* GET -Pengguna menyimpan kasus uji.

Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan akan muncul pada halaman daftar kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan sudah tersimpan

5.2.6. Skenario Pengujian 6: Melihat Detail Kasus Uji yang Telah Dibuat

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU01 dan DU02. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.6.

Tabel 5.6 Skenario Pengujian Melihat Detail Kasus Uji yang Telah Dibuat

Referensi Kasus Penggunaan	DU01, DU02
Nama	Pengujian melihat detail kasus uji yang telah dibuat
Tujuan Pengujian	Melihat detail kasus uji yang telah dibuat
Skenario 1	**Pengguna melihat detail kasus uji yang telah dibuat**
Kondisi Awal	Pengguna berada pada antarmuka *Dashboard*
Data Uji	-
Langkah Pengujian	-Pengguna membuka menu daftar kasus uji -Pengguna memilih kasus uji yang akan dilihat detailnya -Sistem menampilkan detail-detail kasus uji yang dipilih

Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman detail kasus uji -detail kasus uji dapat dilihat oleh pengguna
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dimasukkan telah muncul pada halaman daftar kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Pengguna berada pada antarmuka lihat detail Kasus Uji

5.2.7. Skenario Pengujian 7: Mengubah Detail Kasus Uji yang Telah Dibuat

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU02 dan DU03. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.7.

Tabel 5.7 Skenario Pengujian Merubah Detail Kasus Uji yang Telah Dibuat

Referensi Kasus Penggunaan	DU02, DU03
Nama	Pengujian mengubah detail kasus uji yang telah dibuat
Tujuan Pengujian	Mengubah detail kasus uji yang telah dibuat
Skenario 1	**Pengguna mengubah detail kasus uji yang telah dibuat**
Kondisi Awal	Pengguna berada pada antarmuka lihat detail kasus uji
Data Uji	-Kasus Uji yang digunakan: hasil dari skenario pengujian nomor 2 -data lama: age=20 -data baru: age=21
Langkah Pengujian	-Sistem menampilkan detail-detail kasus uji yang dipilih

	-pengguna menekan tombol untuk mengaktifkan menu pengubahan -sistem menampilkan *form* untuk setiap detil kasus uji beserta tombol untuk menyimpan di sebelah form. -pengguna mengubah detil kasus uji yang akan diubah -pengguna menyimpan perubahan dengan tombol yang terdapat di sebelah kanan *form*
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman detail kasus uji -data detail kasus uji akan berubah
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar kasus uji -data detail kasus uji telah berubah
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji yang dimasukkan sudah tersimpan

5.2.8. Skenario Pengujian 8: Menonaktifkan Kasus Uji

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU02 dan DU04. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.8.

Tabel 5.8 Skenario Pengujian Menonaktifkan Kasus Uji

Referensi Kasus Penggunaan	DU02, DU04
Nama	Pengujian menonaktifkan kasus uji
Tujuan Pengujian	Menonaktifkan kasus uji
Skenario 1	**Pengguna menonaktifkan kasus uji**
Kondisi Awal	-Pengguna berada pada antarmuka daftar kasus uji -Jika kasus uji telah terpasang pada server uji, kasus uji tetap dijalankan -kasus uji terlihat pada daftar kasus uji
Data Uji	-

Langkah Pengujian	-Pengguna menekan tombol hapus yang ada pada kolom kasus uji yang sesuai
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dihapus akan hilang dari daftar kasus uji -Jika kasus uji telah terpasang pada server uji, kasus uji tetap dijalankan
Hasil yang Didapatkan	--sistem mengarahkan pengguna ke halaman daftar kasus uji -data kasus uji yang dihapus hilang dari daftar kasus uji -Jika kasus uji telah terpasang pada server uji, kasus uji tetap dijalankan
Hasil Pengujian	Berhasil
Kondisi Akhir	Kasus uji tidak muncul pada daftar kasus uji

5.2.9. Skenario Pengujian 9: Memasang Kasus Uji Pada Server Uji Sebagai Pengujian

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU05 dan DU06. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.9.

Tabel 5.9 Skenario Pengujian Memasang Kasus Uji pada *Server* Uji Sebagai Pengujian

Referensi Kasus Penggunaan	DU05, DU06
Nama	Memasang Kasus Uji pada *Server* Uji Sebagai Pengujian
Tujuan Pengujian	Menambah pemasangan kasus uji pada server uji sebagai pengujian
Skenario 1	**Pengguna Memasang Kasus Uji pada *Server* Uji Sebagai Pengujian**
Kondisi Awal	Pengguna berada pada antarmuka daftar pemasangan kasus uji

Data Uji	Parameter: Nama Pemasangan: Coba GET dengan Basic Kasus Uji yang digunakan: Kasus Uji pada Skenario pengujian nomor 4 Initiate date= 05/18/2018 Initiate time=00:35 Active=yes Interval=1 menit Endpoint target= postman-echo.com/basic-auth Enable SSL= yes
Langkah Pengujian	-Pengguna membuka menu untuk menambahkan pemasangan kasus uji -Pengguna mengisi parameter-parameter yang ada -Pengguna menyimpan pemasangan kasus uji dengan menekan tombol simpan.
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman daftar pemasangan kasus uji -data pemasangan kasus uji yang dimasukkan akan muncul pada halaman daftar pemasangan kasus uji
Hasil yang Didapatkan	-Sistem mengarahkan pengguna ke halaman daftar pemasangan kasus uji -data pemasangan kasus uji yang dimasukkan telah muncul pada halaman daftar pemasangan kasus uji
Hasil Pengujian	Berhasil
Kondisi Akhir	Pemasangan kasus uji yang dimasukkan sudah tersimpan

5.2.10. Skenario Pengujian 10: Mengubah Detail Pemasangan Kasus Uji

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU05 dan

DU07. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.10.

Tabel 5.10 Skenario Pengujian Mengubah Detail Pemasangan Kasus Uji

Referensi Kasus Penggunaan	DU05, DU07
Nama	Mengubah detail pemasangan kasus uji
Tujuan Pengujian	Mengubah detail pemasangan kasus uji pada server uji
Skenario 1	**Pengguna Mengubah detail pemasangan kasus uji pada *Server* Uji Sebagai Pengujian**
Kondisi Awal	Pengguna berada pada antarmuka detail pemasangan kasus uji
Data Uji	Parameter: Nilai interval lama = 1 menit Nilai interval lama = 2 menit
Langkah Pengujian	-Sistem menampilkan detail-detail kasus uji yang dipilih -pengguna menekan tombol untuk mengaktifkan menu pengubahan -sistem menampilkan *form* untuk setiap detil pemasangan kasus uji beserta tombol untuk menyimpan di sebelah *form*. -pengguna mengubah detail pemasangan kasus uji yang akan diubah -pengguna menyimpan perubahan dengan tombol yang terdapat di sebelah kanan *form*
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman detail pemasangan kasus uji -perubahan detail akan disimpan
Hasil yang Didapatkan	-sistem mengarahkan pengguna ke halaman detail pemasangan kasus uji -perubahan detail telah disimpan
Hasil Pengujian	Berhasil

Kondisi Akhir	Perubahan detail pemasangan kasus uji telah disimpan

5.2.11. Skenario Pengujian 11: Menonaktifkan Pemasangan Pengujian

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DU05 dan DU09. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.11.

Tabel 5.11 Skenario Tabel Pengujian Menonaktifkan Pengujian

Referensi Kasus Penggunaan	DU05, DU09
Nama	Pengujian menonaktifkan pemasangan kasus uji pada server uji
Tujuan Pengujian	Menonaktifkan pengujian
Skenario 1	**Pengguna menonaktifkan pengujian**
Kondisi Awal	Pengguna berada pada antarmuka detail pemasangan kasus uji
Data Uji	-
Langkah Pengujian	-Sistem menampilkan detail-detail kasus uji yang dipilih -pengguna menekan tombol untuk mengaktifkan menu pengubahan -sistem menampilkan *form* untuk setiap detil pemasangan kasus uji beserta tombol untuk menyimpan di sebelah *form*. -pengguna mengubah status aktif. -pengguna menyimpan perubahan dengan tombol yang terdapat di sebelah kanan *form*
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman detail pemasangan kasus uji -pemasangan kasus uji tidak dieksekusi
Hasil yang Didapatkan	-sistem akan mengarahkan pengguna ke halaman detail pemasangan kasus uji -pemasangan kasus uji tidak dieksekusi

Hasil Pengujian	Berhasil
Kondisi Akhir	pemasangan kasus uji tidak dieksekusi

5.2.12. Skenario Pengujian 12: Memberi Akses Pengguna ke *Server* Uji

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan DA01. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.12.

Tabel 5.12 Skenario Tabel Pengujian Memberi Akses Pengguna ke *Server* Uji

Referensi Kasus Penggunaan	DA01
Nama	Pengujian memberi akses pengguna ke *server* uji
Tujuan Pengujian	Memberi akses pengguna ke *server* uji
Skenario 1	**Administrator memberi akses pengguna ke *server* uji**
Kondisi Awal	Administrator berada pada antarmuka pemberian akses pengguna ke server
Data Uji	-
Langkah Pengujian	-Administrator memilih *server* uji mana yang akan digunakan -Sistem akan menampilkan daftar pengguna -Administrator memilih pengguna yang akan diberi akses. -sistem menyimpan perubahan yang dibuat
Hasil yang Diharapkan	-Pengguna akan memiliki akses untuk menggunakan *server* uji
Hasil yang Didapatkan	-pengguna memiliki akses untuk menggunakan server uji
Hasil Pengujian	Berhasil

Kondisi Akhir	Pengguna memiliki akses untuk menggunakan server uji

5.2.13. Skenario Pengujian 13: Melihat Hasil Pengujian Pemasangan Kasus Uji Pada Server Uji

Skenario pengujian ini mencakup pengujian fungsi perangkat lunak untuk menangani kasus penggunaan SU01, SU02, SP01, DU05, dan DU08. Penjelasan lebih lanjut mengenai skenario penggunaan ini terdapat pada tabel 5.13.

Tabel 5.13 Skenario Tabel Pengujian Melihat Hasil Pengujian Pemasangan Kasus Uji Pada Server Uji

Referensi Kasus Penggunaan	SU01, SU02, SP01, DU05, DU08
Nama	Melihat Hasil Pengujian Pemasangan Kasus Uji Pada Server Uji
Tujuan Pengujian	Melihat Hasil Pengujian Pemasangan Kasus Uji Pada Server Uji
Skenario 1	**Pengguna Melihat Hasil Pengujian Pemasangan Kasus Uji Pada Server Uji**
Kondisi Awal	-Pengguna berada pada antarmuka detail pemasangan kasus uji -Scheduler Engine Telah Dinyalakan -Pemasangan Kasus Uji dalam kondisi aktif
Data Uji	-
Langkah Pengujian	-Pengguna memilih pemasangan kasus uji
Hasil yang Diharapkan	-sistem akan mengarahkan pengguna ke halaman detail pemasangan kasus uji -perubahan detail akan disimpan
Hasil yang Didapatkan	-sistem mengarahkan pengguna ke halaman detail pemasangan kasus uji -perubahan detail telah disimpan
Hasil Pengujian	Berhasil
Kondisi Akhir	Perubahan detail pemasangan kasus uji telah disimpan

5.3. Evaluasi Pengujian

Rangkuman mengenai evaluasi pengujian fungsionalitas dapat dilihat pada tabel. Dari dta yang terdapat pada tabel, dapat dilihat bahwa semua kasus penggunaan yang ada telah direpresentasikan dengan paling tidak satu skenario pengujian dan seluruh skenario pengujian berhasil dan program berjalan dengan baik. Sehingga bisa disimpulkan bahwa fungsionalitas program ini bekerja dengan baik.

Tabel 5.14 Rangkuman Hasil Pengujian dengan Skenario Uji

Kode Kasus Penggunaan	Nama	Nomor Skenario pengujian	Hasil
SU01	Mengirim *request* pada *endpoint* sesuai kasus uji	13	Berhasil
SU02	Menyimpan data hasil pengujian	13	Berhasil
SP01	Mengirim perintah pengujian pada *Server* Uji	13	Berhasil
DU01	Menambah kasus uji	1, 2, 3, 4, 5, 6	Berhasil
DU02	Menampilkan daftar kasus uji	6, 7, 8	Berhasil
DU03	Melakukan perubahan pada kasus uji	7	Berhasil
DU04	Nonaktifkan kasus uji	8	Berhasil
DU05	Menampilkan daftar pengujian pada *Server* Uji	9,10,11,13	Berhasil
DU06	Menambahkan pengujian dengan	9	Berhasil

	kasus uji pada *Server* Uji		
DU07	Mengubah pengaturan pengujian pada *Server* Uji	10	Berhasil
DU08	Menampilkan hasil pengujian pada *Server* Uji	13	Berhasil
DU09	Nonaktifkan pengujian pada *Server* Uji	11	Berhasil
DA01	Memberi akses pengguna ke *Server* Uji	12	Berhasil

{Halaman ini sengaja dikosongkan}

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan Tugas Akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian perangkat lunak yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Perangkat lunak dapat melakukan pengujian kinerja pada *web service* dengan protokol REST dan SOAP dengan spesifikasi tertentu yang terdapat pada bagian Batasan masalah pada buku ini.
2. Fungsi *scheduler* pada sistem ini telah dapat bekerja sesuai dengan yang seharusnya.
3. Seluruh skenario pengujian pada bagian aplikasi *Dashboard* berhasil dilaksanakan.
4. Terdapat sedikit keterbatasan pada eksekusi pengujian dikarenakan sistem belum dapat menerima kasus uji dengan parameter selain teks.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Dukungan untuk pembuatan kasus uji berbasis protokol REST yang lainnya seperti PUT dan DELETE dapat ditambahkan pada sistem ini.
2. Dukungan untuk tipe data selain teks untuk parameter pada kasus uji dapat ditambahkan pada sistem ini.
3. Fungsi notifikasi jika terdapat pengujian yang gagal dapat ditambahkan pada sistem ini.

4. Dukungan untuk pembuatan kasus uji berbasis protokol SOAP dengan HTTP *header* yang dapat diatur sendiri oleh pengguna dapat ditambahkan pada sistem ini.

DAFTAR PUSTAKA

[1] V. Beal, "Webopedia," [Online]. Available: https://www.webopedia.com/TERM/H/HTTP.html. [Accessed 25 October 2017].

[2] Wikipedia, "Wikipedia," Wikimedia Fondation, Inc, 24 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Accessed 25 October 2017].

[3] "W3," W3.Org, 1999. [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html. [Accessed 7 January 2017].

[4] S. Haryandi, "Kudo Developers," 15 September 2016. [Online]. Available: https://developers.kudo.co.id/2016/09/15/mengenal-restful-api/. [Accessed 26 October 2017].

[5] W3Schools.com, "w3schools.com," W3.CSS, [Online]. Available: https://www.w3schools.com/php/php_intro.asp. [Accessed 26 October 2017].

[6] Wikipedia, "Wikipedia," Wikimedia Foundation, Inc, 13 October 2017. [Online]. Available: https://en.wikipedia.org/wiki/PHP. [Accessed 26 October 2017].

[7] J. Mueller, "Blog.Smartbear.Com," Smart bear, 8 January 2013. [Online]. Available: https://blog.smartbear.com/apis/understanding-soap-and-rest-basics/. [Accessed 7 January 2017].

[8] "Mkyong.com," Mkyong, [Online]. Available: https://www.mkyong.com/java/how-to-run-a-task-periodically-in-java/. [Accessed 6 January 2017].

[9] Nandu, "StackOverflow," 3 December 2015. [Online]. Available: https://stackoverflow.com/questions/34059737/how-to-get-

server-response-time-using-php. [Accessed 6 January 2017].

[10] G. levin, "Rest Case Blog," 8 November 2016. [Online]. Available: http://blog.restcase.com/restful-api-authentication-basics/. [Accessed 7 January 2017].

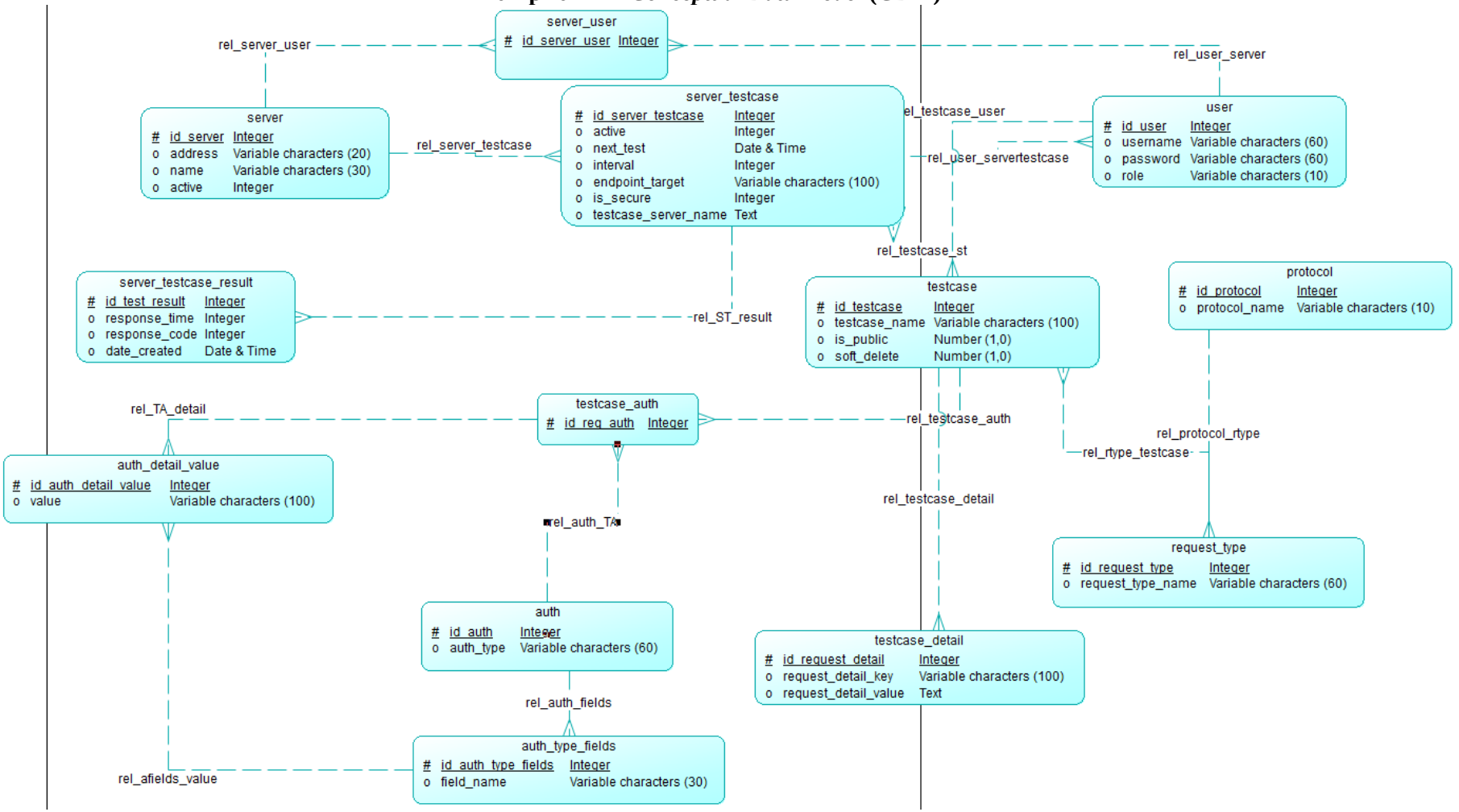
[11] Microsoft, "Microsoft Releases SQL Server 2012 to Help Customers Manage "Any Data, Any Size, Anywhere"," MIcrosoft News Center, 7 March 2012. [Online]. Available: http://www.microsoft.com/Presspass/press/2012/mar12/03-06SQLServer12PR.mspx. [Accessed 5 January 2018].

[12] Microsoft, "SQL Server Documentation," Microsoft, 10 October 2017. [Online]. Available: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation. [Accessed 5 January 2018].

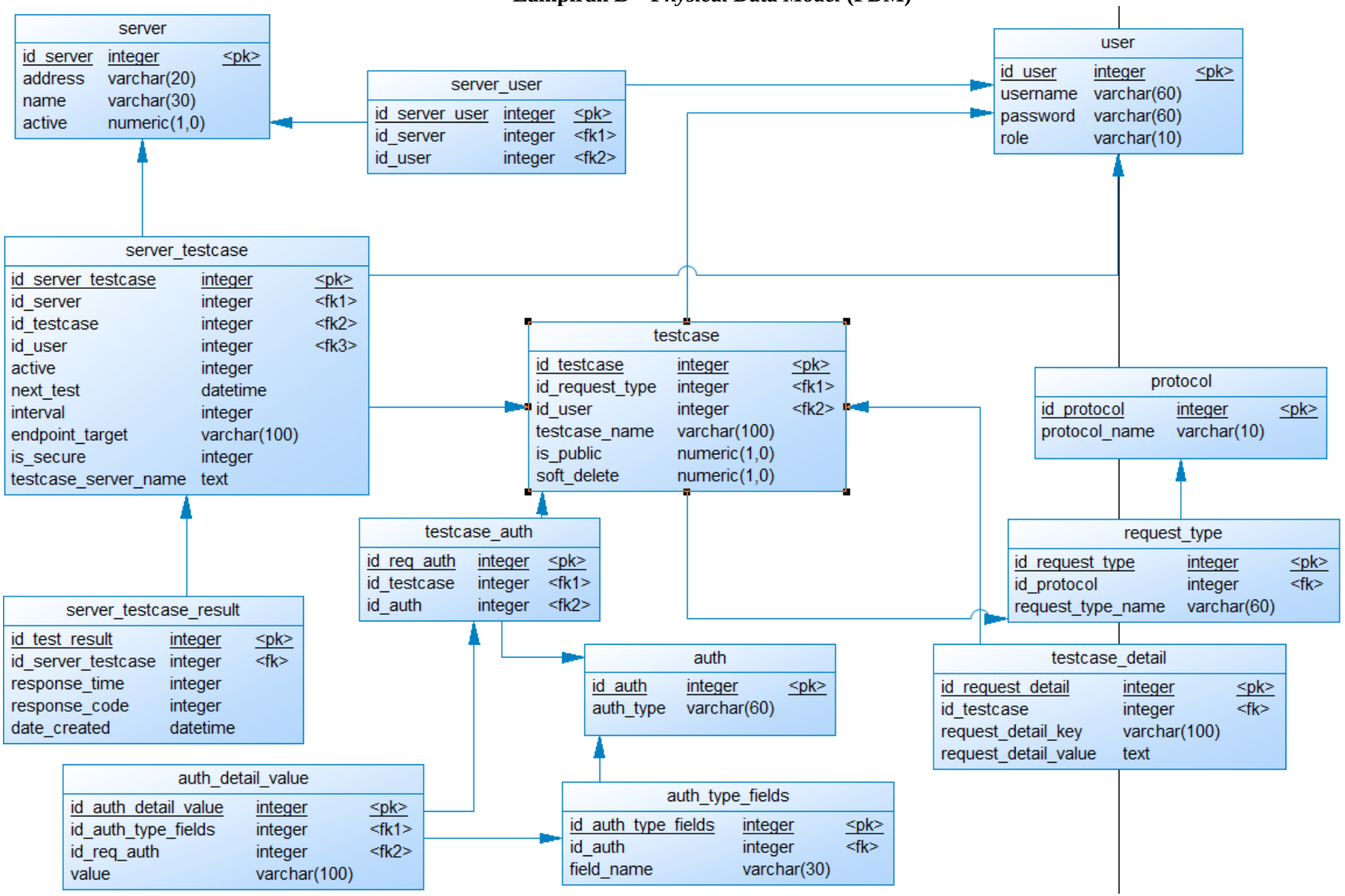
[13] "CodeIgniter," EllisLab, [Online]. Available: https://github.com/bcit-ci/CodeIgniter. [Accessed 29 December 2017].

LAMPIRAN

Lampiran A – *Conceptual Data Model* (CDM)



Lampiran B – *Physical Data Model* (PDM)



BIODATA PENULIS

Penulis, **Alfitrah Nurramadhan Sudirman**, lahir di Yogyakarta, 8 Februari 1997. Penulis menempuh pendidikan sekolah dasar di SD Muhammadiyah Sapen (2003-2008). Melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 5 Yogyakarta (2008-2011) dan selanjutnya di SMA Negeri 1 Yogyakarta (2011-2014). Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Penulis aktif mengerjakan beberapa proyek di antaranya adalah Sistem Informasi Keuangan Institut Pertanian Bogor. Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL). Penulis dapat dihubungi melalui instagram dengan nama pengguna *@fitoalfitrahns* atau melalui surel *fitoalfitrahns@gmail.com*.