BAB VI
PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian terhadap aplikasi web pencarian rute terpendek yang telah dibangun. Proses pengujian dilakukan menggunakan teknik pengujian *White Box (White Box Testing)* dan *Black Box (Black Box Testing).* Strategi pengujian yang diapakai adalah *unit testing, validation testing* dan pengujian sistem.

6.1 *Unit Testing* (Pengujian Unit)

Pada pengujian unit perangkat lunak ini, digunakan teknik pengujian *White Box (White Box Testing)* dengan teknik *Basis Path Testing*. Pada teknik *Basis Path Testing*, proses pengujian dilakukan dengan memodelkan algoritma pada suatu *flow graph*, menentukan jumlah kompleksitas siklomatis *(cyclomatic complexity)*, menentukan sebuah basis set dari jalur independen dan memberikan kasus uji *(test case)* pada setiap basis set yang telah ditentukan. Penulisan laporan skripsi ini hanya dicantumkan hasil pengujian unit untuk algoritma dari beberapa metode (operasi) saja tidak untuk keseluruhan metode.

6.1.1 Pengujian untuk Operasi get_distance

Operasi get_distance merupakan implementasi algoritma untuk mendapatkan jarak terpendek. Operasi (metode) get_distance terdapat di dalam klas prosesJalur. Gambar 6.1 menunjukkan proses pemodelan dalam *flow graph* pada operasi get_distance.

```
NAMA ALGORITMA get_distance

DESKRIPSI
Proses    :
 1. Menginisialisasi variabel $awal = $_POST dan $akhir =
    $_POST merupakan titik awal dan titik akhir
 2. Memanggil variabel $fw sebagai new FloydWarshall
    ($graph, $nodes)                                      } 1
 3. Mencetak variabel $fw->get_distance
```

```
Keluaran :
  Sistem menampilkan jarak dari titik awal menuju titik akhir
```

Gambar 6.1 Algoritma get_distance

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.2 berikut:

Gambar 6.2 *Flowgraph* Operasi get_distance

Sumber: [Pengujian]

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi get_distance menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 0 – 1 + 2
= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.1.

Tabel 6.1 Kasus Uji untuk Pengujian Unit get_distance

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	Menjalankan operasi get_distance	Mendapatkan jarak ter-pendek dari titik awal ke titik akhir	Mendapatkan jarak ter-pendek dari titik awal ke titik akhir

Sumber: [Pengujian]

6.1.2 Pengujian untuk Operasi balasEmail

Operasi balasEmail merupakan implementasi algoritma untuk mengirim email ke *user*. Operasi (metode) balasEmail terdapat di dalam klas hubungi. Gambar 6.3 merupakan proses pemodelan dalam *flow graph* pada algoritma balasEmail.

```
NAMA ALGORITMA balasEmail

DESKRIPSI
Proses    :
 1. Memanggil variabel $tampil untuk menampilkan
    isi pesan sebelumnya
 2. Memanggil method POST action &act=kirimEmail
 3. Input email, subject $ pesan
 4. Input type=submit value=kirim
Keluaran :
 Pesan telah berhasil dikirimkan ke email user
```

1

Gambar 6.3 Algoritma balasEmail

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.4 berikut:

Gambar 6.4 *Flowgraph* Operasi balasEmail

Sumber: [Pengujian]

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi balasEmail menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 0 – 1 + 2

= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.2.

Tabel 6.2 Kasus Uji untuk Pengujian Unit balasEmail

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	Menjalankan operasi balasEmail	*Administrator* membalas pesan yang dikirim *user*	*Administrator* membalas pesan yang dikirim *user*

Sumber: [Pengujian]

6.1.3 Pengujian untuk Operasi input (gedung)

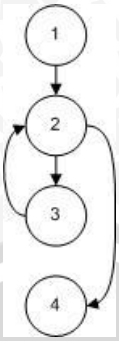
Operasi input merupakan implementasi algoritma untuk menambah data (gedung) baru. Operasi (metode) input terdapat di dalam klas aksi. Gambar 6.5 merupakan proses pemodelan dalam *flow graph* pada algoritma input.

```
NAMA ALGORITMA input

DESKRIPSI
Proses   :
 1. Memanggil $module=='gedung', $act=='input'        } 1
 2. Jika id_gedung & nm_gedung 'empty'          (2)
    a. echo alert 'harap isi semua data'        } 3
    b. refresh halaman input gedung
 3. Jika tidak maka,
    a. Insert into table, dengan menggunakan method $_POST
    b. Input data                                       } 4
    c. Input type=submit value=kirim

Keluaran :
 Data baru berhasil ditambahkan kedalam database
```

Gambar 6.5 Algoritma input

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.6 berikut:



Gambar 6.6 *Flowgraph* Operasi input

Sumber: [Pengujian]

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi input menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 4 – 4 + 2
= 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1 – 2 – 3 – 2 – 4

Jalur 2 : 1 – 2 – 4

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.3.

Tabel 6.3 Kasus Uji untuk Pengujian Unit input

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	Memeriksa input data	Keluar peringatan jika data kosong	Sistem menampilkan peringatan ke *user*
2	Menjalankan operasi input	Data baru dapat ditambahkan ke dalam *database*	Data baru dapat ditambahkan ke dalam *database*

Sumber: [Pengujian]

6.1.4 Pengujian untuk Operasi delete

Operasi delete merupakan implementasi algoritma untuk menghapus data yang telah ada. Operasi (metode) delete terdapat di dalam klas aksi. Gambar 6.7 merupakan proses pemodelan dalam *flow graph* pada algoritma update.

```
NAMA ALGORITMA delete

DESKRIPSI
Proses    :
 1. Memanggil $module dan $act=='hapus'
 2. Delete from .$module where id_.$module
 3. Menghapus data berdasarkan id_data
Keluaran :
 Data yang ada berhasil dihapus
```

1

Gambar 6.7 Algoritma delete

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.8 berikut:

Gambar 6.8 *Flowgraph* Operasi delete

Sumber: [Pengujian]

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi delete menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node).*

V(G) = E – N + 2

= 0 – 1 + 2

= 1

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.4.

Tabel 6.4 Kasus Uji untuk Pengujian Unit *delete*

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	Menjalankan operasi delete	Data yang ada berhasil dihapus	Data yang ada berhasil dihapus

Sumber: [Pengujian]

6.1.5 Pengujian untuk Operasi cekLogin

Operasi cekLogin merupakan implementasi algoritma untuk menjalankan proses login. Operasi (metode) cekLogin terdapat di dalam klas admin. Gambar 6.9 merupakan proses pemodelan dalam *flow graph* pada algoritma cekLogin.

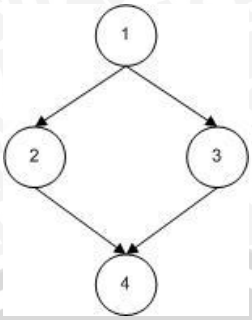
```
NAMA ALGORITMA cekLogin

DESKRIPSI
Proses   :
  1. Memanggil variabel $login select from table user
  2. Memanggil variable $ketemu=mysql_num_row($login)
  3. Apabila username dan password ditemukan
     a. if (ketemu > 0) session_start dan
        session_register
     b. Memanggil method $_SESSION[..]=$r[..]
     c. Go to home page
  4. Jika username dan password tidak ditemukan
     a. echo "Username dan password tidak benar"
     b. echo "Silakan ulangi lagi"

Keluaran :
 Algoritma berhasil memeriksa nilai input dari user
```

1 2 3 4

Gambar 6.9 Algoritma cekLogin

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.10 berikut:



Gambar 6.10 *Flowgraph* Operasi cekLogin

Sumber: [Pengujian]

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi cekLogin menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – 4 + 2
= 4 – 4 + 2
= 2

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1 – 2 – 4

Jalur 2 : 1 – 3 – 4

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.5.

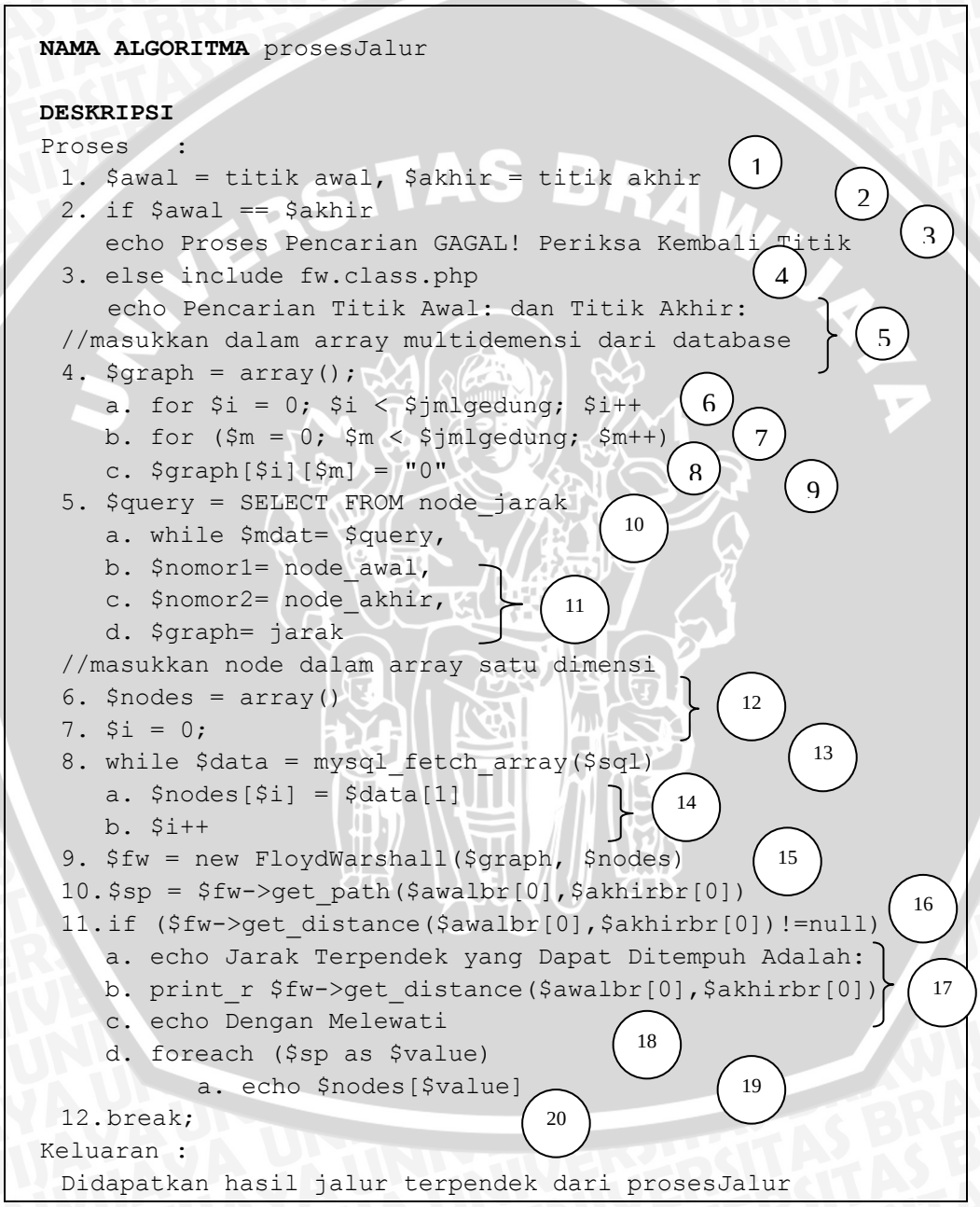
Tabel 6.5 Kasus Uji untuk Pengujian Unit *cekLogin*

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	*User* memasukkan data yang benar	Sistem masuk ke halaman utama	Sistem masuk ke halaman utama
2	*User* memasukkan data yang salah	Sistem menampilkan peringatan kepada *user*	Sistem menampilkan peringatan kepada *user*

Sumber: [Pengujian]

6.1.6 Pengujian untuk Operasi prosesJalur

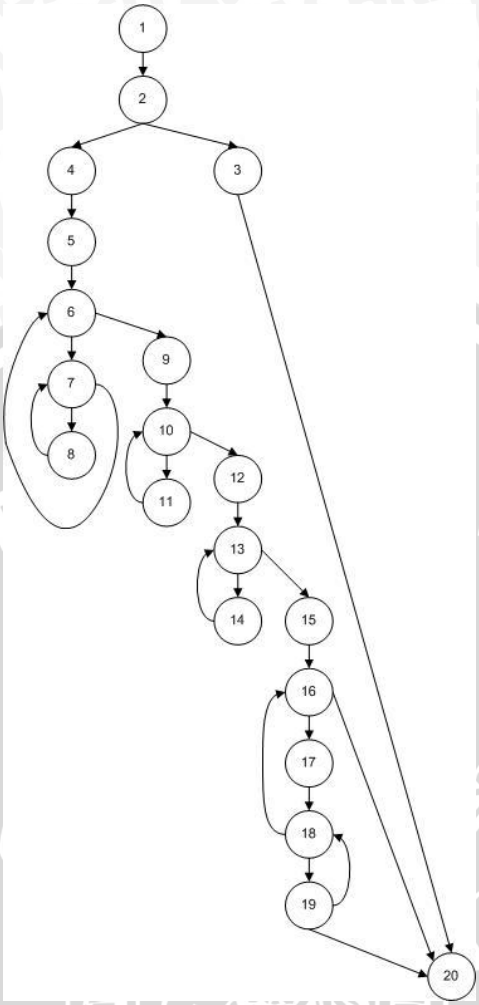
Operasi prosesJalur merupakan implementasi algoritma untuk menjalankan proses pencarian jalur. Operasi (metode) prosesJalur terdapat di dalam klas jalur. Gambar 6.11 merupakan proses pemodelan dalam *flow graph* pada algoritma prosesJalur.



Gambar 6.11 Algoritma prosesJalur

Sumber: [Pengujian]

Dari algortima tersebut didapatkan *flowgraph* seperti pada Gambar 6.10 berikut:



Gambar 6.12 *Flowgraph* Operasi prosesJalur

Pemodelan ke dalam *flowgraph* yang telah dilakukan terhadap operasi prosesJalur menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* melalui persamaan V(G) = E – N + 2, dimana V(G) merupakan jumlah kompleksitas siklomatis, E merupakan sisi (garis penghubung antar *node*) dan N merupakan jumlah simpul *(node)*.

V(G) = E – N + 2
= 27 – 20 + 2
= 9

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu ditentukan satu buah basis set dari jalur independent yaitu:

Jalur 1 : 1 – 2 – 3 – 20

Jalur 2 : 1 – 2 – 4 – 5 – 6 – 7 – 8 – 7 – 6 – 9 – …

Jalur 3 : 1 – 2 – 4 – 5 – 6 – 7 – 6 – 9 – 10 – …

Jalur 4 : 1 – 2 – 4 – 5 – 6 – 9 – 10 – 11 – 10 – 12 – 13 – 14 – …

Jalur 5 : 1 – … – 6 – 9 – 10 – 12 – 13 – 14 – 13 – 15 – 16 – 17 – …

Jalur 6 : 1 – … – 6 – 9 – 10 – 12 – 13 – 15 – 16 – 20

Jalur 7 : 1 – … – 6 – 9 – 10 – 12 – 13 – 15 – 16 – 17 – 18 – 19 – 20

Jalur 8 : 1 – … – 6 – 9 – 10 – 12 – 13 – 15 – 16 – 17 – 18 –16 – 20

Jalur 9 : 1 – … – 6 – 9 – 10 – 12 – 13 – 15 – 16 – 17 – 18 –19 – 18 – 19 – 20

Penentuan kasus uji untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji dijelaskan pada Tabel 6.6.

Tabel 6.6 Kasus Uji untuk Pengujian Integrasi prosesJalur

Jalur	Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapat
1	Memeriksa titik awal = titik akhir	Sistem menampilan peringatan	Sistem menampilkan peringatan kepada *user*
2	Memeriksa penambahan gedung	Sistem dapat memeriksa proses penambahan gedung	Sistem dapat memeriksa proses penambahan gedung
3	Memasukkan database kedalam array	Data graf pada database dapat di masukkan kedalam array	Data graf pada database berhasil di masukkan kedalam array
4	Memanggil query node_jarak	Data berupa node_awal, node_akhir, dan jarak berhasil dipanggil	Data berupa node_awal, node_akhir, dan jarak berhasil dipanggil
5	Memasukkan node dalam array	Node baru ditambahkan kedalam array	Node baru telah ditambahkan kedalam array
6	Memeriksa node dan graph tidak null	Sistem melakukan perhitungan jarak	Sistem berhasil melakukan perhitungan jarak
7	Menghitung jalur terpendek	Sistem menghitung jarak terpendek	Sistem dapat menghitung jarak terpendek

8	Menampilkan jarak total	Sistem menampilkan perhitungan jarak terpendek	Sistem menampilkan perhitungan jarak terpendek
9	Menampilkan node yang dilalui	Sistem menampilakn node yang dapat dilalui	Sistem menampilakn node yang dapat dilalui

Sumber: [Pengujian]

6.2 *Validation Testing*

Pengujian fungsional dilakukan untuk mencoba apakah fungsi-fungsi dasar aplikasi berjalan sebagaimana mestinya. Uji coba fungsionalitas meliputi semua *use case* yang ada pada bab perancangan.

6.2.1 Pengujian Melihat Info Jurusan

Hasil uji coba proses pengujian melihat informasi jurusan disajikan pada Tabel 6.7

Tabel 6.7 UJI_SRS_001_01

ID	**UJI_SRS_001_01**
Nama	Pengujian melihat info jurusan
Tujuan uji coba	Menguji fungsi melihat info jurusan
Precondition	Layar menampilkan menu utama
Skenario	***User* memilih melihat info jurusan**
Input	Mengarahkan dan menekan kursor *mouse* pada salah satu gedung
Output yang diharapkan	Layar akan menampilkan segala informasi mengenai jurusan atau fakultas tersebut
Hasil uji coba	Valid
Postcondition	Pada layar ditampilkan informasi mengenai jurusan atau fakultas di UB

Sumber: [Pengujian]

6.2.2 Pengujian Mencari Rute Terpendek

Hasil uji coba proses pengujian mencari rute terpendek antar jurusan atau fakultas disajikan pada Tabel 6.8

Tabel 6.8 UJI_SRS_001_02

ID	**UJI_SRS_001_02**
Nama	Pengujian mencari rute terpendek
Tujuan uji coba	Menguji fungsi pencarian rute terpendek
Precondition	Layar menampilkan menu info jalur
Skenario	***User* memilih mencari rute terpendek**
Input	Memilih node_awal dan node_akhir sebagai kode dari jurusan atau fakultas
Output yang diharapkan	Layar akan menampilkan hasil perhitungan rute terpendek beserta rute yang bisa dilewati
Hasil uji coba	Valid
Postcondition	Pada layar ditampilkan hasil perhitungan rute terpendek antar jurusan atau fakultas di UB beserta rute yang bisa dilewati

Sumber: [Pengujian]

6.2.3 Pengujian Mengirim Pesan

Hasil uji coba proses mengirim pesan disajikan pada Tabel 6.9

Tabel 6.9 UJI_SRS_001_03

ID	**UJI_SRS_001_03**
Nama	Pengujian mengirim pesan
Tujuan uji coba	Menguji fungsi mengirim pesan
Precondition	Layar menampilkan menu hubungi
Skenario	***User* mengisikan pesan kemudian dikirim kepada *administrator***
Input	Mengisi *field* nama, alamat email, dan isi pesan
Output yang diharapkan	Layar akan menampilkan pemberitahuan bahwa pesan telah berhasil dikirimkan
Hasil uji coba	Valid
Postcondition	Pada layar ditampilkan pemberitahuan bahwa pesan telah berhasil dikirimkan kepada

	administrator

Sumber: [Pengujian]

6.2.4 Pengujian *Login Administrator*

Hasil uji coba proses pengujian *login* bagi *administrator* disajikan pada Tabel 6.10

Tabel 6.10 UJI_SRS_002_01

ID	**UJI_SRS_002_01**
Nama	Pengujian *Login administrator*
Tujuan uji coba	Menguji fungsi *login* bagi *administrator*
Precondition	Layar menampilkan menu utama halaman *admin*
Skenario	***Administrator* memasukkan *username* dan *password* untuk dapat masuk kedalam sistem**
Input	Memasukkan *username* dan *password*
Output yang diharapkan	*Administrator* dapat masuk ke dalam sistem
Hasil uji coba	Valid
Postcondition	*Administrator* berhasil masuk kedalam sistem

Sumber: [Pengujian]

6.2.5 Pengujian *Logout Administrator*

Hasil uji coba proses pengujian *login* bagi *administrator* disajikan pada Tabel 6.11

Tabel 6.11 UJI_SRS_002_02

ID	**UJI_SRS_002_02**
Nama	Pengujian *Logout administrator*
Tujuan uji coba	Menguji fungsi *logout* bagi *administrator*
Precondition	Layar menampilkan halaman *administrator*
Skenario	***Administrator* menekan tombol logout untuk dapat keluar dari sistem**
Input	Menekan tombol *Logout*
Output yang diharapkan	*Administrator* dapat keluar dari system

Hasil uji coba	Valid
Postcondition	*Administrator* berhasil keluar dari system

Sumber: [Pengujian]

6.2.6 Pengujian Tambah Akun *Administrator*

Hasil uji coba proses pengujian tambah akun baru bagi *administrator* disajikan pada Tabel 6.12

Tabel 6.12 UJI_SRS_002_03

ID	**UJI_SRS_002_03**
Nama	Pengujian tambah akun
Tujuan uji coba	Menguji fungsi tambah akun
Precondition	Layar menampilkan halaman manajemen *user*
Skenario	***Administrator* menambahkan *admin* baru**
Input	Mengisi data *admin* baru
Output yang diharapkan	*Administrator* baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	*Administrator* baru berhasil ditambahkan

Sumber: [Pengujian]

6.2.7 Pengujian Edit Akun *Administrator*

Hasil uji coba pengujian mengedit akun *admin* disajikan pada Tabel 6.13

Tabel 6.13 UJI_SRS_002_04

ID	**UJI_SRS_002_04**
Nama	Pengujian edit akun
Tujuan uji coba	Menguji fungsi edit akun
Precondition	Layar menampilkan halaman manajemen *user*
Skenario	***Administrator* mengedit data *admin* yang telah ada**
Input	Mengisi data *admin* baru
Output yang diharapkan	Data *Administrator* baru berhasil ditambahkan

Hasil uji coba	Valid
Postcondition	Data *Administrator* berhasil diedit

Sumber: [Pengujian]

6.2.8 Pengujian Hapus Akun *Administrator*

Hasil uji proses pengujian hapus akun *admin* disajikan pada Tabel 6.14

Tabel 6.14 UJI_SRS_002_05

ID	**UJI_SRS_002_05**
Nama	Pengujian hapus akun
Tujuan uji coba	Menguji fungsi hapus akun
Precondition	Layar menampilkan halaman manajemen *user*
Skenario	***Administrator* menghapus data *admin* yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	*Administrator* yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	*Administrator* yang ada berhasil dihapus

Sumber: [Pengujian]

6.2.9 Pengujian Tambah Modul

Hasil uji coba proses pengujian tambah modul disajikan pada Tabel 6.15

Tabel 6.15 UJI_SRS_002_06

ID	**UJI_SRS_002_06**
Nama	Pengujian tambah modul
Tujuan uji coba	Menguji fungsi tambah modul
Precondition	Layar menampilkan halaman manajemen manajemen modul
Skenario	***Administrator* menambahkan modul baru**
Input	Mengisi data modul baru
Output yang diharapkan	Modul baru berhasil ditambahkan
Hasil uji coba	Valid

Postcondition	Modul baru berhasil ditambahkan

Sumber: [Pengujian]

6.2.10 Pengujian Edit Modul

Hasil uji proses pengujian mengedit modul disajikan pada Tabel 6.16

Tabel 6.16 UJI_SRS_002_07

ID	**UJI_SRS_002_07**
Nama	Pengujian edit modul
Tujuan uji coba	Menguji fungsi edit modul
Precondition	Layar menampilkan halaman manajemen modul
Skenario	***Administrator* mengedit data modul yang telah ada**
Input	Mengisi data modul baru
Output yang diharapkan	Data modul baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Data modul berhasil diedit

Sumber: [Pengujian]

6.2.11 Pengujian Hapus Modul

Hasil uji coba proses pengujian hapus modul disajikan pada Tabel 6.17

Tabel 6.17 UJI_SRS_002_08

ID	**UJI_SRS_002_08**
Nama	Pengujian hapus modul
Tujuan uji coba	Menguji fungsi hapus modul
Precondition	Layar menampilkan halaman manajemen modul
Skenario	***Administrator* menghapus modul yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	Modul yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	Modul yang ada berhasil dihapus

Sumber: [Pengujian]

6.2.12 Pengujian Tambah Jalur

Hasil uji coba proses pengujian tambah jalur disajikan pada Tabel 6.18

Tabel 6.18 UJI_SRS_002_09

ID	**UJI_SRS_002_09**
Nama	Pengujian tambah jalur
Tujuan uji coba	Menguji fungsi tambah jalur
Precondition	Layar menampilkan halaman jalur
Skenario	***Administrator* menambahkan jalur baru**
Input	Mengisi data jalur baru
Output yang diharapkan	Jalur baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Jalur baru berhasil ditambahkan

Sumber: [Pengujian]

6.2.13 Pengujian Edit Jalur

Hasil uji coba proses pengujian mengedit jalur disajikan pada Tabel 6.19

Tabel 6.16 UJI_SRS_002_10

ID	**UJI_SRS_002_10**
Nama	Pengujian edit jalur
Tujuan uji coba	Menguji fungsi edit jalur
Precondition	Layar menampilkan halaman jalur
Skenario	***Administrator* mengedit data jalur yang telah ada**
Input	Mengisi data jalur baru
Output yang diharapkan	Data jalur baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Data jalur berhasil diedit

Sumber: [Pengujian]

6.2.14 Pengujian Hapus Jalur

Hasil uji coba proses pengujian hapus jalur disajikan pada Tabel 6.20

Tabel 6.20 UJI_SRS_002_11

ID	**UJI_SRS_002_11**
Nama	Pengujian hapus jalur
Tujuan uji coba	Menguji fungsi hapus jalur
Precondition	Layar menampilkan halaman manajemen jalur
Skenario	***Administrator* menghapus jalur yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	Jalur yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	Jalur yang ada berhasil dihapus

Sumber: [Pengujian]

6.2.15 Pengujian Tambah Gedung

Hasil uji proses pengujian tambah gedung disajikan pada Tabel 6.21.

Tabel 6.21 UJI_SRS_002_12

ID	**UJI_SRS_002_12**
Nama	Pengujian tambah gedung
Tujuan uji coba	Menguji fungsi tambah gedung
Precondition	Layar menampilkan halaman gedung
Skenario	***Administrator* menambahkan gedung baru**
Input	Mengisi data gedung baru
Output yang diharapkan	Gedung baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Gedung baru berhasil ditambahkan

Sumber: [Pengujian]

6.2.16 Pengujian Edit Gedung

Hasil uji proses pengujian mengedit gedung disajikan pada Tabel 6.22.

Tabel 6.22 UJI_SRS_002_13

ID	**UJI_SRS_002_13**

Nama	Pengujian edit gedung
Tujuan uji coba	Menguji fungsi edit gedung
Precondition	Layar menampilkan halaman gedung
Skenario	***Administrator* mengedit data gedung yang telah ada**
Input	Mengisi data gedung baru
Output yang diharapkan	Data gedung baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Data gedung berhasil diedit

Sumber: [Pengujian]

6.2.17 Pengujian Hapus Gedung

Hasil uji coba proses pengujian hapus gedung disajikan pada Tabel 6.23

Tabel 6.23 UJI_SRS_002_14

ID	**UJI_SRS_002_14**
Nama	Pengujian hapus gedung
Tujuan uji coba	Menguji fungsi hapus gedung
Precondition	Layar menampilkan halaman manajemen gedung
Skenario	***Administrator* menghapus gedung yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	Gedung yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	Gedung yang ada berhasil dihapus

Sumber: [Pengujian]

6.2.18 Pengujian Tambah Peta

Hasil uji coba proses pengujian tambah peta disajikan pada Tabel 6.24

Tabel 6.24 UJI_SRS_002_15

ID	**UJI_SRS_002_15**
Nama	Pengujian tambah peta

Tujuan uji coba	Menguji fungsi tambah peta
Precondition	Layar menampilkan halaman peta
Skenario	***Administrator* menambahkan peta baru**
Input	Mengisi data peta baru
Output yang diharapkan	Peta baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Peta baru berhasil ditambahkan

Sumber: [Pengujian]

6.2.19 Pengujian Edit Peta

Hasil uji coba proses pengujian mengedit peta disajikan pada Tabel 6.25

Tabel 6.25 UJI_SRS_002_16

ID	**UJI_SRS_002_16**
Nama	Pengujian edit peta
Tujuan uji coba	Menguji fungsi edit peta
Precondition	Layar menampilkan halaman peta
Skenario	***Administrator* mengedit data peta yang telah ada**
Input	Mengisi data peta baru
Output yang diharapkan	Data peta baru berhasil ditambahkan
Hasil uji coba	Valid
Postcondition	Data peta berhasil diedit

Sumber: [Pengujian]

6.2.20 Pengujian Hapus Peta

Hasil uji coba proses pengujian hapus gedung disajikan pada Tabel 6.26

Tabel 6.26 UJI_SRS_002_17

ID	**UJI_SRS_002_17**
Nama	Pengujian hapus peta
Tujuan uji coba	Menguji fungsi hapus peta

Precondition	Layar menampilkan halaman manajemen peta
Skenario	***Administrator* menghapus peta yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	Peta yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	Peta yang ada berhasil dihapus

Sumber: [Pengujian]

6.2.21 Pengujian Balas Pesan

Hasil uji coba proses pengujian balas pesan disajikan pada Tabel 6.27

Tabel 6.27 UJI_SRS_001_18

ID	**UJI_SRS_002_18**
Nama	Pengujian balas pesan
Tujuan uji coba	Menguji fungsi membalas pesan kepada *user*
Precondition	Layar menampilkan menu hubungi kami
Skenario	***Administrator* membalas pesan kepada *user***
Input	Menekan email *user* dan mengisikan pesan yang akan dikirim
Output yang diharapkan	Layar akan menampilkan informasi bahwa pesan telah berhasil dikirim
Hasil uji coba	Valid
Postcondition	Pada layar ditampilkan informasi bahwa pesan telah berhasil dikirim

Sumber: [Pengujian]

6.2.22 Pengujian Hapus Pesan

Hasil uji coba proses pengujian hapus pesan disajikan pada Tabel 6.28

Tabel 6.28 UJI_SRS_002_19

ID	**UJI_SRS_002_19**
Nama	Pengujian hapus pesan
Tujuan uji coba	Menguji fungsi hapus pesan

Precondition	Layar menampilkan halaman hubungi kami
Skenario	***Administrator* menghapus pesan yang ada**
Input	Menekan tombol Hapus
Output yang diharapkan	Pesan yang ada berhasil dihapus
Hasil uji coba	Valid
Postcondition	Pesan yang ada berhasil dihapus

Sumber: [Pengujian]

6.3 Pengujian Akurasi Sistem

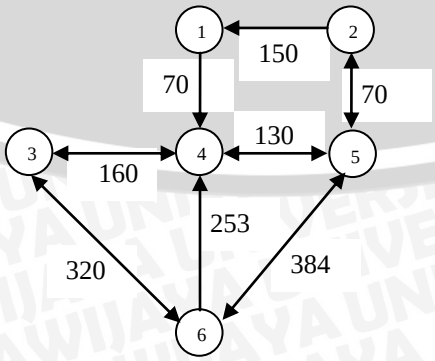
Aplikasi web pencarian rute terpendek antar jurusan atau fakultas di Universitas Brawijaya ini telah diuji dengan menggunakan beberapa kasus. Pengujian yang terakhir yaitu menggunakan pengujian akurasi dari implementasi algoritma Floyd-warshall. Data yang diuji berjumlah 20 data. Prosedur pengujiannya adalah memasukkan titik awal dan titik akhir, kemudian sistem menghasilkan hasil prediksi. Hasil prediksi tersebut kemudian dicocokkan kesesuaiannya dengan prediksi dari perhitungan manual. Perhitungan untuk pengujian akurasi dapat dijabarkan dengan *accuracy* sebagai berikut:

$$accuracy = \frac{N_c}{N} \quad (6\text{-}1)$$

Dimana:

- N_c : *Number of positive instances covered by rule*
- N : *Number of instances covered by rule*

Pengujian Akurasi dilakukan pada 6 node gedung jurusan atau fakultas yang terdapat di kampus Universitas Brawijaya.



Gambar 6.13 Sebagian *node* kampus UB

Sumber: [Pengujian]

Keterangan gambar:

1. Gedung Widyaloka
2. Gedung Fakultas THP
3. Gedung Jurusan Fisika
4. Gedung Fakultas Ilmu Bahasa
5. Gedung Perpustakaan
6. Gedung Gazebo

Hasil pengujian akurasi dari 20 data yang diuji adalah sebagai berikut:

Tabel 6.29 Hasil Pengujian Akurasi

No	Awal	Akhir	Jalur (Aplikasi)	Jalur (Manual)	Hasil
1	THP	FIB	THP-Widyaloka-FIB	THP-Widyaloka-FIB	P
2	THP	Fisika	THP-Widyaloka-FIB-Fisika	THP-Widyaloka-FIB-Fisika	P
3	THP	Gazebo	THP-Perpustakaan-Gazebo	THP-Perpustakaan-Gazebo	P
4	Widyaloka	Fisika	Widyaloka-FIB-Fisika	Widyaloka-FIB-Fisika	P
5	Widyaloka	Gazebo	Widyaloka-FIB-Fisika-Gazebo	Widyaloka-FIB-Fisika-Gazebo	P
6	Widyaloka	Perpustakaan	Widyaloka-FIB-Perpustakaan	Widyaloka-FIB-Perpustakaan	P
7	Gazebo	Perpustakaan	Gazebo-FIB-Perpustakaan	Gazebo-FIB-Perpustakaan	P
8	Gazebo	Fisika	Gazebo-Fisika	Gazebo-Fisika	P
9	Gazebo	THP	Gazebo-FIB-Perpustakaan-THP	Gazebo-FIB-Perpustakaan-THP	P
10	Gazebo	FIB	Gazebo-FIB	Gazebo-FIB	P
11	Perpustakaan	FIB	Perpustakaan-THP-Widyaloka-FIB	Perpustakaan-THP-Widyaloka-FIB	P
12	Perpustakaan	Fisika	Perpustakaan-THP-Widyaloka-FIB-Fisika	Perpustakaan-THP-Widyaloka-FIB-Fisika	P
13	Perpustakaan	Widyaloka	Perpustakaan-THP-Widyaloka	Perpustakaan-THP-Widyaloka	P
14	FIB	Gazebo	FIB-Fisika-Gazebo	FIB-Fisika-Gazebo	P
15	FIB	Perpustakaan	FIB-Perpustakaan	FIB-Perpustakaan	P

16	FIB	THP	FIB-Perpustakaan-THP	FIB-Perpustakaan-THP	P
17	Fisika	FIB	Fisika-FIB	Fisika-FIB	P
18	Fisika	THP	Fisika-FIB-Perpustakaan-THP	Fisika-FIB-Perpustakaan-THP	P
19	Fisika	Perpustakaan	Fisika-FIB-Perpustakaan	Fisika-FIB-Perpustakaan	P
20	Fisika	Gazebo	Fisika-Gazebo	Fisika-Gazebo	P

Sumber: [Pengujian]

Keterangan :

- P: jika sistem mampu memprediksi rute terpendek yang telah dimasukkan oleh *user* dan hasilnya sama dengan hasil perhitungan manual.
- N: jika sistem tidak mampu memprediksi rute terpendek yang telah dimasukkan oleh *user* dan hasilnya berbeda dengan hasil perhitungan manual.

Jika P adalah positif, dan N adalah negatif, maka:

$$accuracy = \frac{N_c}{N}$$

$$= 20 / 20 = 100\%$$

Berdasarkan hasil pengujian akurasi dengan 20 data dihasilkan akurasi 100 %.