BAB IV
PERANCANGAN

Pada bab ini membahas mengenai perancangan perangkat lunak. Perancangan yang dilakukan meliputi dua tahap. Tahap analisis kebutuhan dan perancangan mengacu pada pemodelan *waterfall*. Proses analisis kebutuhan dilakukan pada tahap pertama, setelah tahap analisis kebutuhan sudah dipahami dan dapat didefinisikan dengan baik, tahap yang kedua adalah proses perancangan perangkat lunak. Tahap analisis kebutuhan terdiri dari dua langkah yaitu membuat daftar kebutuhan *user* dan menggunakan pemodelan *use case diagram* untuk menggambarkan kebutuhan tersebut. Sedangkan proses perancangan perangkat lunak mempunyai empat langkah, yaitu perancangan *class diagram, sequence diagram,* perancangan basis data, dan perancangan antarmuka web menggambarkan perencanaan tampilan web yang akan dibuat.

4.1 Analisis Kebutuhan (*Requirement Analysis*)

Tahap analisis kebutuhan ini diawali dengan identifikasi aktor yang terlibat dengan sistem, penjabaran kebutuhan dan kemudian memodelkannya ke dalam suatu *use case diagram. Use case diagram* bertujuan untuk menggambarkan kebutuhan-kebutuhan fungsional yang harus disediakan oleh *software* sistem informasi agar dapat memecahkan permasalahan yang dihadapi oleh pengguna. Tahap ini dilakukan dengan memodelkan kebutuhan sistem ke dalam *use case diagram*.

Proses analisis dilakukan dengan acuan hasil pengumpulan, pemahaman, dan penetapan kebutuhan-kebutuhan dari aplikasi yang akan dibangun. Analisis kebutuhan dibagi menjadi dua tahap yaitu identifikasi aktor-aktor dan penjabaran daftar kebutuhan. Analisis kebutuhan ini ditujukan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

4.1.1 Identifikasi Aktor

Tahap ini mempunyai tujuan untuk melakukan identifikasi terhadap aktor yang akan berinteraksi dengan sistem. Tabel 4.1 memperlihatkan dua buah

aktor beserta penjelasannya masing-masing yang merupakan hasil dari proses identifikasi aktor.

Tabel 4.1 Deskripsi Aktor

Aktor	Deskripsi Aktor
User	*User* merupakan semua aktor yang menggunakan fasilitas aplikasi sistem informasi pencarian rute terpendek. Aktor ini memiliki hak akses yaitu memilih menu yang tersedia, mencari informasi lokasi jurusan, mencari rute terpendek antar jurusan/ fakultas, dan mengirim pesan kepada *administrator*.
Administrator	*Administrator* merupakan aktor yang memantau dan me-*maintenance* sistem. Aktor ini memilik hak akses yaitu menambah, menghapus, meng-*update* rute, gedung, peta yang terdapat di sistem, dan memberikan informasi jurusan.

Sumber: [Perancangan]

4.1.2 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan nonfungsional yang terdiri dari sebuah kolom yang menguraikan kebutuhan yang harus disediakan oleh sistem, dan pada kolom yang lain akan menunjukkan nama *use case* yang akan menyediakan fungsionalitas masing-masing kebutuhan tersebut. Pada daftar kebutuhan akan dispesifikasikan menjadi dua yaitu spesifikasi kebutuhan fungsional *user* dan spesifikasi kebutuhan fungsional *administrator*. Spesifikasi kebutuhan fungsional *user* ditunjukkan pada Tabel 4.2.

Tabel 4.2 Daftar Kebutuhan *User*

SRS No	Tabel Kebutuhan	*Use Case*
SRS_01_01	Sistem harus mampu menyediakan fasilitas bagi *user* untuk mencari informasi lokasi jurusan atau fakultas di UB.	Melihat informasi jurusan
SRS_01_02	Sistem harus mampu menyediakan fasilitas bagi *user* untuk mencari rute terpendek antar jurusan atau fakultas di UB.	Mencari rute terpendek
SRS_01_03	Sistem harus mampu menyediakan fasilitas untuk *user* agar dapat mengirim pesan.	Mengirim pesan

Sumber: [Perancangan]

Spesifikasi kebutuhan fungsional *administrator* ditunjukkan pada Tabel 4.3

Tabel 4.3 Daftar Kebutuhan *Administrator*

SRS No	Tabel Kebutuhan	*Use Case*
SRS_02_01	Sistem harus mampu memberikan fasilitas untuk *login*, sehingga hanya *administrator* yang dapat mengelola sistem.	*Login*
SRS_02_02	Sistem harus mampu memberikan fasilitas untuk *login*, sehingga *administrator* yang telah *login* dapat keluar dari sistem.	*Logout*
SRS_02_03	Sistem harus menyediakan fasilitas untuk menambah akun *administrator* baru.	Tambah akun
SRS_02_04	Sistem harus menyediakan fasilitas untuk meng-edit akun *administrator*.	Edit akun
SRS_02_05	Sistem harus menyediakan fasilitas untuk menghapus akun *administrator*.	Hapus akun
SRS_02_06	Sistem harus menyediakan fasilitas *administrator* untuk menambah modul.	Tambah modul
SRS_02_07	Sistem harus menyediakan fasilitas *administrator* untuk meng-edit modul.	Edit modul
SRS_02_08	Sistem harus menyediakan fasilitas *administrator* untuk menghapus modul.	Hapus modul
SRS_02_09	Sistem harus menyediakan fasilitas untuk menambah rute yang terdapat di UB.	Tambah rute
SRS_02_10	Sistem harus menyediakan fasilitas untuk meng-edit rute yang terdapat di UB.	Edit rute
SRS_02_11	Sistem harus menyediakan fasilitas untuk menghapus rute yang terdapat di UB.	Hapus rute
SRS_02_12	Sistem harus menyediakan fasilitas untuk menambah gedung yang terdapat di UB.	Tambah gedung
SRS_02_13	Sistem harus menyediakan fasilitas untuk meng-edit gedung yang terdapat di UB.	Edit gedung
SRS_02_14	Sistem harus menyediakan fasilitas untuk menghapus gedung yang terdapat di UB.	Hapus gedung

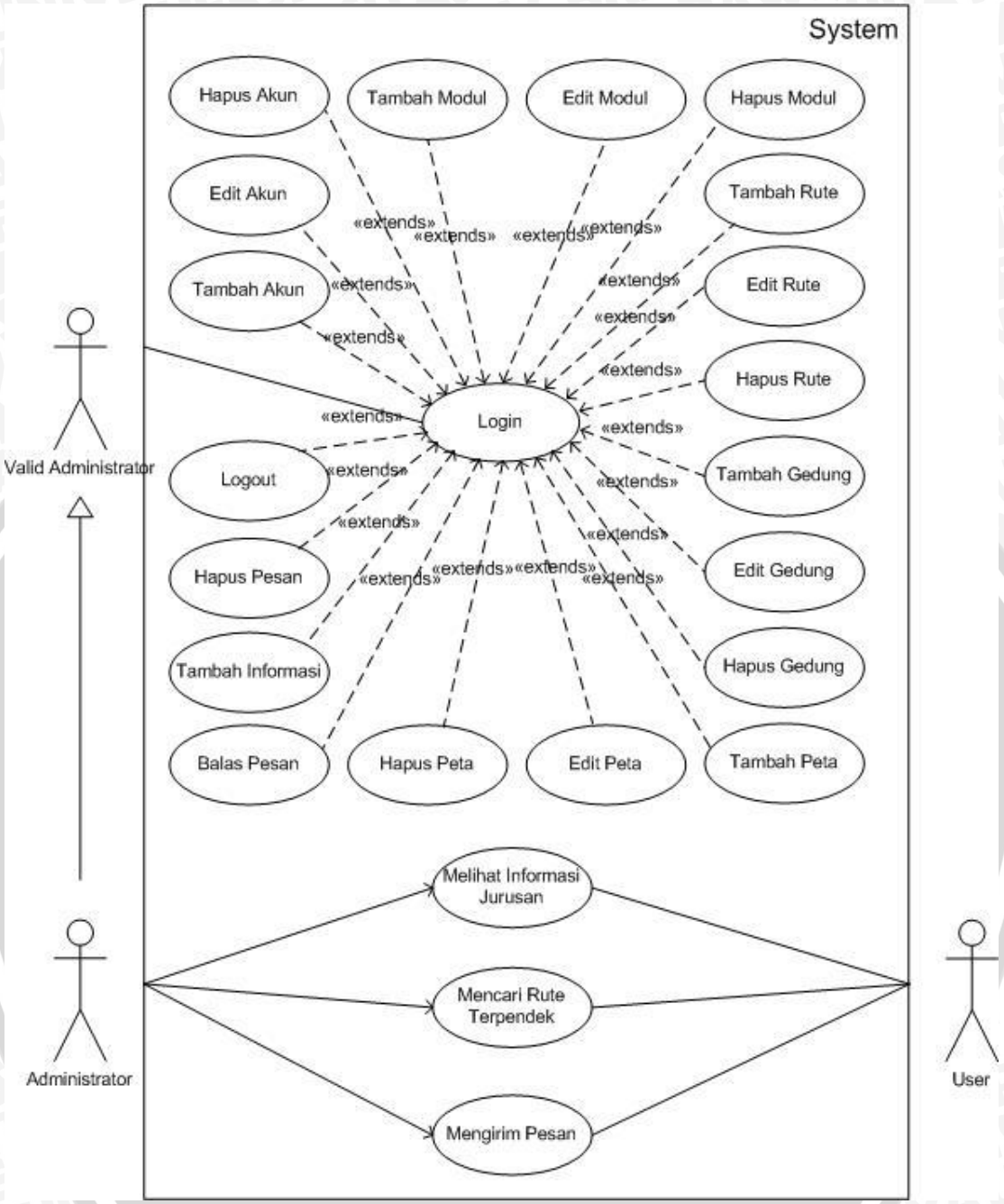
SRS_02_15	Sistem harus menyediakan fasilitas untuk menambah peta animasi rute yang terdapat di UB.	Tambah peta
SRS_02_16	Sistem harus menyediakan fasilitas untuk meng-edit peta animasi rute yang terdapat di UB.	Edit peta
SRS_02_17	Sistem harus menyediakan fasilitas untuk menghapus peta animasi rute yang terdapat di UB.	Hapus peta
SRS_02_18	Sistem harus menyediakan fasilitas untuk membalas pesan yang dikirim *user*.	Balas pesan
SRS_02_19	Sistem harus menyediakan fasilitas untuk menghapus pesan yang dikirim *user*.	Hapus pesan
SRS_02_20	Sistem harus bisa menambah informasi jurusan atau fakultas yang terdapat di UB.	Tambah informasi jurusan

Sumber: [Perancangan]

4.1.3 *Use Case Diagram*

Kebutuhan-kebutuhan fungsional yang diperlukan oleh pengguna dan harus disediakan oleh sistem akan dimodelkan pada diagram *use case*. Masing-masing *use case diagram* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. *Use case* merupakan fungsionalitas dari sistem yang diinisialisasi oleh aktor. Secara keseluruhan sistem informasi pencarian rute terpendek ini dibagi menjadi dua yaitu *use case diagram* untuk *user* dan *use case diagram* untuk *administrator*.

Pada *user* terdapat tiga buah *use case*, yaitu *use case* Melihat informasi jurusan, Mencari rute terpendek, dan Mengirim pesan. *Administrator* biasa juga berperan sebagai *user* yang bisa melihat informasi dan mencari rute terpendek. Sedangkan untuk *valid administrator* terdapat 20 buah *use case*, yaitu *use case Login*, *Logout*, Tambah akun, Edit akun, Hapus akun, Tambah modul, Edit modul, Hapus modul, Tambah rute, Edit rute, Hapus rute, Tambah gedung, Edit gedung, Hapus gedung, Tambah peta, Edit peta, Hapus peta, Balas pesan, Hapus pesan, dan Tambah informasi jurusan. Gambar 4.1 merupakan *use case diagram* keseluruhan sistem.



Gambar 4.1 *Use Case Diagram* sistem

Sumber: [Perancangan]

Secara lebih mendetail, masing-masing *use case* yang terdapat pada diagram *use case*, dijabarkan dalam skenario *use case*. Di dalam skenario *use case*, akan diberikan uraian nama *use case*, aktor yang berhubungan dengan *use case* tersebut, tujuan dari *use case*, deskripsi global tentang *use case*, pra-kondisi yang harus dipenuhi dan pos-kondisi yang diharapkan setelah berjalannya

fungsional *use case*. Selain itu juga akan diberikan ulasan yang berkaitan dengan tanggapan dari sistem atas suatu aksi yang diberikan oleh aktor (aliran utama), serta kejadian alternatif yang akan terjadi jika suatu kondisi tidak bisa terpenuhi (aliran alternatif).

Kebutuhan fungsional yang harus disediakan oleh sistem adalah kebutuhan untuk melihat informasi jurusan/ fakultas yang terdapat di kampus. Kebutuhan tersebut direpresentasikan oleh *Use Case* Melihat Informasi Jurusan. Tabel 4.4 merupakan skenario *use case* melihat informasi jurusan.

Tabel 4.4 Skenario *Use Case* Melihat Informasi Jurusan

Use Case	Melihat informasi jurusan
Aktor	*User*
Tujuan	Untuk mencari informasi fakultas atau jurusan yang dibutuhkan *user*.
Deskripsi	Fasilitas yang diberikan sistem untuk mencari informasi fakultas atau jurusan yang dibutuhkan oleh *user.*
Pra-kondisi	*User* membuka menu 'Beranda' pada sistem informasi web pencarian rute terpendek.
Pos-kondisi	*User* akan mendapatkan segala informasi mengenai fakultas atau jurusan yang dibutuhkannya.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. Pada menu utama, *user* memilih fakultas atau jurusan yang dibutuhkan.	2. Melakukan proses pemeriksaan permintan *user*. Setelah itu sistem akan menampilkan data pada halaman menu utama.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mencari rute terpendek antar jurusan/ fakultas yang terdapat di kampus. Kebutuhan tersebut direpresentasikan oleh *Use Case* Mencari Rute Terpendek. Tabel 4.5 merupakan skenario *use case* mencari rute terpendek.

Tabel 4.5 Skenario *Use Case* Mencari Rute Terpendek

Use Case	Mencari rute terpendek
Aktor	*User*
Tujuan	Untuk mencari rute terpendek antar fakultas atau jurusan yang dibutuhkan *user*.
Deskripsi	Fasilitas ini digunakan untuk mencari rute terpendek antar jurusan atau fakultas yang dibutuhkan oleh *user* untuk mencari lokasi jurusan yang tersebut.
Pra-kondisi	*User* membuka menu 'Info Jalur' pada sistem informasi web pencarian rute terpendek.
Pos-kondisi	*User* akan mendapatkan informasi rute terpendek yang dapat ditempuh untuk menuju sebuah lokasi yang dibutuhkan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. Pada menu rute, *user* memasukkan titik awal dan titik akhir jurusan atau fakultas yang dibutuhkan.	2. Melakukan proses pemeriksaan permintan *user*. Setelah itu sistem akan menampilkan informasi rute yang dapat ditempuh untuk menuju sebuah lokasi yang dibutuhkan.
Aliran Alternatif 1 : Eksepsi jika titik yang dipilih sama.	
	3. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 1.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengirim saran, kritik, maupun pertanyaan kepada *administrator*. Kebutuhan tersebut direpresentasikan oleh *Use Case* Mengirim Pesan. Tabel 4.6 merupakan skenario *use case* mengirim pesan.

Tabel 4.6 Skenario *Use Case* Mengirim Pesan

Use Case	Mengirim pesan
Aktor	*User*
Tujuan	Untuk mengirim pesan kepada pihak *administrator*.

Deskripsi	Fasilitas ini digunakan untuk mengirim pesan yang bisa berisi saran, kritik, maupun pertanyaan kepada *administrator*.
Pra-kondisi	*User* membuka menu 'Hubungi Kami' pada sistem informasi web pencarian rute terpendek.
Pos-kondisi	*User* telah berhasil mengirimkan pesan kepada *administrator*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *User* memilih menu 'Hubungi Kami'	2. Sistem menampilkan halaman 'Hubungi Kami'. Halaman 'Hubungi Kami' dalam bentuk textfield untuk mengisikan nama, alamat email, subjek, dan isi pesan. Terdapat tombol Kirim.
3. *User* mengisikan nama, alamat email, subjek, dan isi pesan. Kemudain menekan tombol Kirim.	4. Sistem melakukan pemeriksaan terhadap isi textfield, jika semua data telah diisi maka pesan telah berhasil dikirim kepada *administrator*.
Aliran Alternatif 1 : Eksepsi jika ada textfield yang belum diisi.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk *login* kedalam sistem bagi *administrator*. Kebutuhan tersebut direpresentasikan oleh *Use Case Login*. Tabel 4.7 merupakan skenario *use case login*.

Tabel 4.7 Skenario *Use Case Login*

Use Case	*Login*
Aktor	*Administrator*
Tujuan	Untuk melakukan autentivikasi pemakai
Deskripsi	Fasilitas yang disediakan sistem untuk masuk ke sistem.

Pra-kondisi	*Administrator* masuk ke halaman *login*
Pos-kondisi	*Administrator* berhasil melakukan proses *login*, sistem menampilkan halaman utama.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu *login.*	2. Sistem menampilkan halaman *login*. Halaman *login* dalam bentuk textfield untuk mengisikan *username* dan *password*. Terdapat tombol masuk.
3. *Administrator* mengisikan *user-name* dan *password* dan menekan tombol masuk.	4. Sistem melakukan pemeriksaan karakter terhadap *username* dan *password*, selanjutnya sistem melakukan pemeriksaan kedalam basisdata, jika *username* dan *password* benar maka akan masuk ke menu utama.
Aliran Alternatif 1 : Eksepsi jika *username* dan *password* salah.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mendaftarkan akun *administrator* baru. Kebutuhan tersebut direpresentasikan oleh *Use Case* Tambah Akun. Tabel 4.8 merupakan skenario *use case* tambah akun.

Tabel 4.8 Skenario *Use Case* Tambah Akun

Use Case	Tambah akun
Aktor	*Administrator*
Tujuan	Untuk melakukan registrasi bagi akun baru.
Deskripsi	Fasilitas yang disediakan sistem untuk memungkinkan penambahan *administrator*.

Pra-kondisi	Sistem menampilkan halaman manajemen *user*.
Pos-kondisi	*Administrator* baru telah terdaftar dalam sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu manajemen *user*.	2. Sistem menampilkan halaman manajemen *user*.
3. *Administrator* mengisi form tambah *user* dengan mengisikan *username* dan *password*.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka administrator baru akan terdaftar .
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak sesuai.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengedit akun *administrator*. Kebutuhan tersebut direpresentasikan oleh *Use Case* Edit Akun. Tabel 4.9 merupakan skenario *use case* edit akun.

Tabel 4.9 Skenario *Use Case* Edit Akun

Use Case	Edit akun
Aktor	*Administrator*
Tujuan	Untuk mengganti *password* dari *administrator*.
Deskripsi	Fasilitas yang disediakan sistem untuk memungkinkan *administrator* dalam mengganti *password*.
Pra-kondisi	Sistem menampilkan halaman manajemen *user*.
Pos-kondisi	*Password* lama telah diganti.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu manajemen *user*.	2. Sistem menampilkan halaman manajemen *user*.
3. *Administrator* mengganti *password*	4. Sistem melakukan pemeriksaan isian

dengan mengisikan kembali *username*, *password* lama, dan mengisikan kembali *password* baru kemudian menekan tombol simpan.	data, jika data yang ditambahkan benar dan sesuai maka password telah diganti ke dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus akun *administrator*. Kebutuhan tersebut direpresentasikan oleh *use case* Hapus Akun. Tabel 4.10 merupakan skenario *use case* Hapus Akun.

Tabel 4.10 Skenario *Use Case* Hapus Akun

Use Case	Hapus akun
Aktor	*Administrator*
Tujuan	Untuk menghapus *administrator* yang ada.
Deskripsi	Fasilitas yang disediakan sistem untuk memungkinkan menghapus *administrator*.
Pra-kondisi	Sistem menampilkan halaman manajemen *user*.
Pos-kondisi	*Administrator* lama telah dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu manajemen *user*.	2. Sistem menampilkan halaman manajemen *user*.
3. *Administrator* menghapus akun yang ingin dihapus dengan menekan tombol aksi ‘Hapus’.	4. Sistem menghapus data akun yang tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menambah modul baru. Kebutuhan tersebut direpresentasikan oleh *use case* Tambah Modul. Tabel 4.11 merupakan skenario *use case* Tambah Modul.

Tabel 4.11 Skenario *Use Case* Tambah Modul

Use Case	Tambah Modul

Aktor	*Administrator*
Tujuan	Untuk melakukan penambahan modul baru.
Deskripsi	Fasilitas yang disediakan sistem untuk menambahkan modul baru yang terdapat pada sistem.
Pra-kondisi	Sistem menampilkan halaman manajemen modul.
Pos-kondisi	Modul baru telah berhasil ditambahkan ke dalam basisdata.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu Manajemen Modul.	2. Sistem menampilkan halaman Modul.
3. *Administrator* menambahkan modul baru dengan mengisi nama modul, link, status, dan urutan. Kemudian menekan tombol Simpan.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah ditambahkan ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengedit modul jika terdapat perubahan. Kebutuhan tersebut direpresentasikan oleh *Use Case* Edit Modul. Tabel 4.12 merupakan skenario *use case* edit modul.

Tabel 4.12 Skenario *Use Case* Edit Modul

Use Case	Edit modul
Aktor	*Administrator*
Tujuan	Untuk mengedit modul jika ada perubahan.
Deskripsi	Fasilitas yang disediakan sistem untuk mengedit modul yang telah ada.
Pra-kondisi	Sistem menampilkan halaman manajemen modul.
Pos-kondisi	Modul lama telah berhasil di edit.
Aliran Utama	

Aksi dari Aktor	Tanggapan dari sistem
1. *Administrator* memilih menu Manajemen Modul.	2. Sistem menampilkan halaman modul.
3. *Administrator* mengedit modul dengan memilih dan mengisi nama modul, link, status, dan urutan. Kemudian menekan tombol Simpan.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah di-*update* ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus modul yang ada. Kebutuhan tersebut direpresentasikan oleh *Use Case* Hapus Modul. Tabel 4.13 merupakan skenario *use case* hapus modul.

Tabel 4.13 Skenario *Use Case* Hapus Modul

Use Case	Hapus Modul
Aktor	*Administrator*
Tujuan	Untuk melakukan penghapusan modul.
Deskripsi	Fasilitas yang disediakan sistem untuk menghapus modul yang telah ada.
Pra-kondisi	Sistem menampilkan halaman manajemen modul.
Pos-kondisi	Modul lama telah berhasil dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu Manajemen Modul.	2. Sistem menampilkan halaman modul.
3. *Administrator* menghapus modul dengan memilih modul yang akan dihapus kemudian menekan tombol hapus.	4. Sistem menghapus data modul yang tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menambah rute baru jika ada perubahan di kampus. Kebutuhan tersebut direpresentasikan oleh *Use Case* Tambah Rute. Tabel 4.14 merupakan skenario *use case* tambah rute.

Tabel 4.14 Skenario *Use Case* Tambah Rute

Use Case	Tambah Rute	
Aktor	*Administrator*	
Tujuan	Untuk melakukan penambahan rute jika ada perubahan.	
Deskripsi	Fasilitas yang disediakan sistem untuk menambahkan rute baru yang terdapat di kampus.	
Pra-kondisi	Sistem menampilkan halaman jalur.	
Pos-kondisi	Rute baru telah berhasil ditambahkan ke dalam basisdata.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari sistem**
1. *Administrator* memilih menu jalur.		2. Sistem menampilkan halaman jalur.
3. *Administrator* menambahkan rute baru dengan mengisi titik awal, titik akhir, dan jarak. Kemudian menekan tombol tambah.		4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah ditambahkan ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.		
		5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengedit rute jika terdapat perubahan. Kebutuhan tersebut direpresentasikan oleh *Use Case* Edit Rute. Tabel 4.15 merupakan skenario *use case* edit rute.

Tabel 4.15 Skenario Use Case Edit Rute

Use Case	Edit rute
Aktor	*Administrator*

Tujuan	Untuk mengedit rute jika ada perubahan.
Deskripsi	Fasilitas yang disediakan sistem untuk mengedit rute yang telah ada.
Pra-kondisi	Sistem menampilkan halaman jalur.
Pos-kondisi	Rute lama telah berhasil di-*update*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu jalur.	2. Sistem menampilkan halaman jalur.
3. *Administrator* mengedit rute dengan memilih dan mengisi titik awal, titik akhir, dan jarak. Kemudian menekan tombol edit.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah di-*update* ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus rute yang ada. Kebutuhan tersebut direpresentasikan oleh *Use Case* Hapus Rute. Tabel 4.16 merupakan skenario *use case* hapus rute.

Tabel 4.16 Skenario *Use Case* Hapus Rute

Use Case	Hapus Rute
Aktor	*Administrator*
Tujuan	Untuk melakukan penghapusan rute.
Deskripsi	Fasilitas yang disediakan sistem untuk menghapus rute.
Pra-kondisi	Sistem menampilkan halaman jalur.
Pos-kondisi	Rute lama telah berhasil dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu jalur.	2. Sistem menampilkan halaman jalur.
3. *Administrator* menghapus rute	4. Sistem menghapus data rute yang

dengan memilih rute yang akan dihapus kemudian menekan tombol hapus.	tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menambah gedung baru. Kebutuhan tersebut direpresentasikan oleh *Use Case* Tambah Gedung. Tabel 4.17 merupakan skenario *use case* tambah gedung.

Tabel 4.17 Skenario *Use Case* Tambah Gedung

Use Case	Tambah Gedung	
Aktor	*Administrator*	
Tujuan	Untuk melakukan penambahan gedung jika ada perubahan.	
Deskripsi	Fasilitas yang disediakan sistem untuk menambahkan gedung baru yang terdapat di kampus.	
Pra-kondisi	Sistem menampilkan halaman gedung.	
Pos-kondisi	Gedung baru telah berhasil ditambahkan ke dalam basisdata.	
Aliran Utama		
Aksi dari Aktor		**Tanggapan dari sistem**
1. *Administrator* memilih menu gedung.		2. Sistem menampilkan halaman gedung.
3. *Administrator* menambahkan gedung baru dengan mengisi id gedung, dan nama gedung. Kemudian menekan tombol Simpan.		4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah ditambahkan ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.		
		5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengedit gedung jika terdapat perubahan. Kebutuhan

tersebut direpresentasikan oleh *Use Case* Edit Gedung. Tabel 4.18 merupakan skenario *use case* edit gedung.

Tabel 4.18 Skenario *Use Case* Edit Gedung

Use Case	Edit gedung
Aktor	*Administrator*
Tujuan	Untuk mengedit gedung jika ada perubahan.
Deskripsi	Fasilitas yang disediakan sistem untuk mengedit gedung yang telah ada.
Pra-kondisi	Sistem menampilkan halaman gedung.
Pos-kondisi	Gedung lama telah berhasil di-*update*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu gedung.	2. Sistem menampilkan halaman gedung.
3. *Administrator* mengedit gedung dengan mengisi nama gedung. Kemudian menekan tombol edit.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah di-*update* ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus gedung yang ada. Kebutuhan tersebut direpresentasikan oleh *Use Case* Hapus Gedung. Tabel 4.19 merupakan skenario *use case* hapus gedung.

Tabel 4.19 Skenario *Use Case* Hapus Gedung

Use Case	Hapus gedung
Aktor	*Administrator*
Tujuan	Untuk melakukan penghapusan gedung.
Deskripsi	Fasilitas yang disediakan sistem untuk menghapus gedung

	yang telah ada.
Pra-kondisi	Sistem menampilkan halaman gedung.
Pos-kondisi	Gedung lama telah berhasil dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu gedung.	2. Sistem menampilkan halaman gedung.
3. *Administrator* menghapus gedung dengan memilih gedung yang akan dihapus kemudian menekan tombol hapus.	4. Sistem menghapus data gedung yang tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menambah peta baru. Kebutuhan tersebut direpresentasikan oleh *Use Case* Tambah Peta. Tabel 4.20 merupakan skenario *use case* tambah peta.

Tabel 4.20 Skenario *Use Case* Tambah Peta

Use Case	Tambah Peta
Aktor	*Administrator*
Tujuan	Untuk melakukan penambahan peta.
Deskripsi	Fasilitas yang disediakan sistem untuk menambahkan peta baru yang terdapat di kampus.
Pra-kondisi	Sistem menampilkan halaman peta.
Pos-kondisi	Peta baru telah berhasil ditambahkan ke dalam basisdata.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu peta.	2. Sistem menampilkan halaman peta.
3. *Administrator* menambahkan peta baru dengan mengisi titik awal, titik akhir, *browse file*, dan keterangan. Kemudian menekan tombol Simpan.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah ditambahkan ke dalam basisdata.

Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk mengedit peta jika terdapat perubahan. Kebutuhan tersebut direpresentasikan oleh *Use Case* Edit Peta. Tabel 4.21 merupakan skenario *use case* edit peta.

Tabel 4.21 Skenario *Use Case* Edit Peta

Use Case	Edit peta
Aktor	*Administrator*
Tujuan	Untuk mengedit peta jika ada perubahan.
Deskripsi	Fasilitas yang disediakan sistem untuk mengedit peta yang telah ada.
Pra-kondisi	Sistem menampilkan halaman peta.
Pos-kondisi	Peta lama telah berhasil di-*update*.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu peta.	2. Sistem menampilkan halaman peta.
3. *Administrator* mengedit peta dengan memilih dan mengisi titik awal, titik akhir, *browse file*, dan keterangan. Kemudian menekan tombol edit.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka data telah di-*update* ke dalam basisdata.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus peta yang ada. Kebutuhan tersebut

direpresentasikan oleh *Use Case* Hapus Peta. Tabel 4.22 merupakan skenario *use case* hapus peta.

Tabel 4.22 Skenario *Use Case* Hapus Peta

Use Case	Hapus peta
Aktor	*Administrator*
Tujuan	Untuk melakukan penghapusan peta.
Deskripsi	Fasilitas yang disediakan sistem untuk menghapus peta yang telah ada.
Pra-kondisi	Sistem menampilkan halaman peta.
Pos-kondisi	Peta lama telah berhasil dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu peta.	2. Sistem menampilkan halaman peta.
3. *Administrator* menghapus peta dengan memilih peta yang akan dihapus kemudian menekan tombol hapus.	4. Sistem menghapus data peta yang tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk membalas pesan yang dkirimkan oleh *user*. Kebutuhan tersebut direpresentasikan oleh *Use Case* Balas Pesan. Tabel 4.23 merupakan skenario *use case* balas pesan.

Tabel 4.23 Skenario *Use Case* Balas Pesan

Use Case	Balas pesan
Aktor	*Administrator*
Tujuan	Untuk membalas pesan yang dikirim *user*.
Deskripsi	Fasilitas yang disediakan sistem untuk *administrator* agar dapat membalas pesan dari *user*.
Pra-kondisi	Sistem masuk ke halaman hubungi.
Pos-kondisi	Pesan yang ada telah berhasil dibalas oleh *administrator*.
Aliran Utama	

Aksi dari Aktor	Tanggapan dari sistem
1. *Administrator* membuka halaman hubungi.	2. Sistem menampilkan halaman hubungi.
3. *Administrator* membalas pesan yang dikirimkan *user*.	4. Sistem melakukan pemeriksaan isian data, jika data yang ditambahkan benar dan sesuai maka pesan telah berhasil dikirimkan ke user.
Aliran Alternatif 1 : Eksepsi jika data yang diisikan tidak lengkap.	
	5. Sistem menampilkan pesan kesalahan dan akan kembali ke langkah 2.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menghapus pesan yang ada. Kebutuhan tersebut direpresentasikan oleh *Use Case* Hapus Pesan. Tabel 4.24 merupakan skenario *use case* hapus pesan.

Tabel 4.24 Skenario *Use Case* Hapus Pesan

Use Case	Hapus pesan
Aktor	*Administrator*
Tujuan	Untuk melakukan penghapusan pesan.
Deskripsi	Fasilitas yang disediakan sistem untuk menghapus pesan yang dikirimkan oleh *user*.
Pra-kondisi	Sistem menampilkan halaman hubungi.
Pos-kondisi	Pesan lama telah berhasil dihapus.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* memilih menu hubungi.	2. Sistem menampilkan halaman hubungi.
3. *Administrator* menghapus pesan dengan memilih pesan yang akan dihapus kemudian menekan tombol hapus.	4. Sistem menghapus data pesan yang tersimpan dalam basisdata.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk menambah informasi jurusan/ fakultas yang terdapat di kampus. Kebutuhan tersebut direpresentasikan oleh *Use Case* Tambah Informasi Jurusan. Tabel 4.25 merupakan skenario *use case* tambah informasi jurusan.

Tabel 4.25 Skenario *Use Case* Tambah Informasi Jurusan

Use Case	Tambah informasi jurusan
Aktor	*Administrator*
Tujuan	Untuk menambahkan informasi jurusan atau fakultas yang terdapat di kampus.
Deskripsi	Fasilitas yang disediakan sistem untuk melihat informasi jurusan atau fakultas di kampus.
Pra-kondisi	*Administrator* masuk ke halaman *file* peta kampus.
Pos-kondisi	Informasi baru telah berhasil ditambahkan.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* membuka *file* peta kampus yang berupa *file* flash.	2. Sistem menampilkan peta kampus.
3. *Administrator* menambahkan informasi jurusan atau fakultas.	4. Sistem menampilkan informasi jurusan atau fakultas yang terbaru yang berupa *file* flash.

Sumber: [Perancangan]

Kebutuhan fungsional selanjutnya yang harus disediakan oleh sistem adalah kebutuhan untuk keluar dari sistem. Kebutuhan tersebut direpresentasikan oleh *use case Logout*. Tabel 4.26 merupakan skenario *use case Logout*.

Tabel 4.26 Skenario *Use Case Logout*

Use Case	*Logout*
Aktor	*Administrator*
Tujuan	Untuk keluar dari autentivikasi *user*.
Deskripsi	Fasilitas yang disediakan oleh sistem agar aktor yang telah *login* dapat keluar dari sistem.

Pra-kondisi	Sistem dijalankan pada kondisi *login*.
Pos-kondisi	*Administrator* berhasil melakukan proses *logout* dan berhasil keluar dari sistem.
Aliran Utama	
Aksi dari Aktor	**Tanggapan dari sistem**
1. *Administrator* menekan menu *logout*.	2. Sistem menghapus *session*.

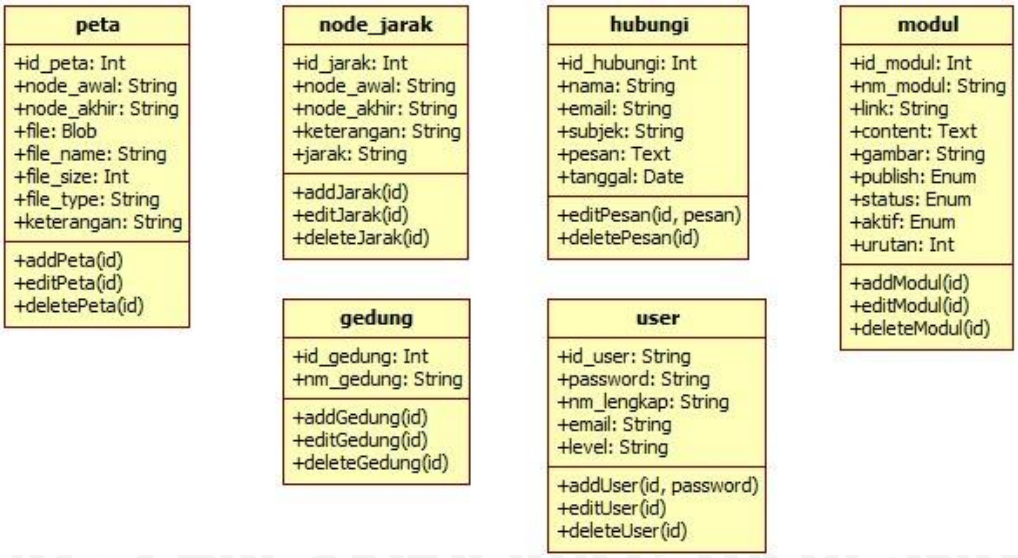
Sumber: [Perancangan]

4.2 Perancangan Perangkat Lunak

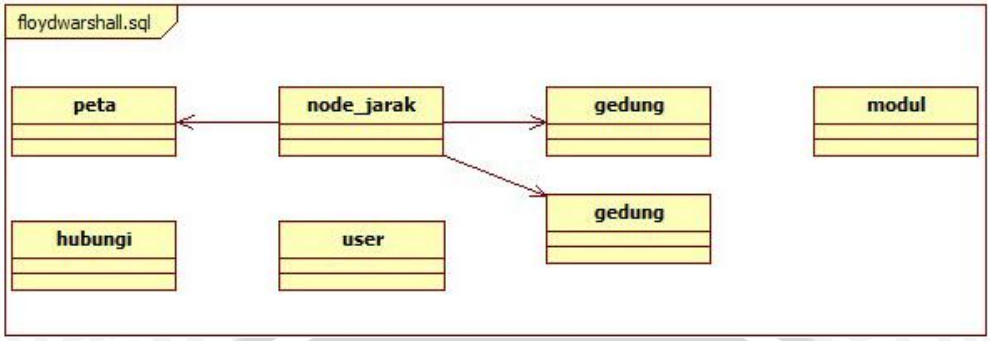
Perancangan perangkat lunak dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan. Perancangan perangkat lunak menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan subsistem dilakukan dengan mengidentifikasi kelas-kelas dan *interface-interface* yang dibutuhkan yang dimodelkan dalam *class diagram*. Hubungan interaksi antar elemen (objek) yang telah diidentifikasi, dimodelkan dalam *Sequence Diagram* yang menggambarkan interaksi antar objek yang disusun dalam urutan waktu.

4.2.1 Diagram Klas *(Class Diagram)*

Diagram klas memberikan gambaran pemodelan elemen-elemen klas yang membentuk sebuah sistem perangkat lunak mulai dari atribut-atribut serta method-methodnya dan hubungannya dengan klas-klas lain dalam sebuah sistem. Gambar *Class* diagram ditunjukkan oleh Gambar 4.2.



Gambar 4.2 Diagram Klas Sistem
Sumber: [Perancangan]



Gambar 4.3 Relasi Antar Klas

Sumber: [Perancangan]

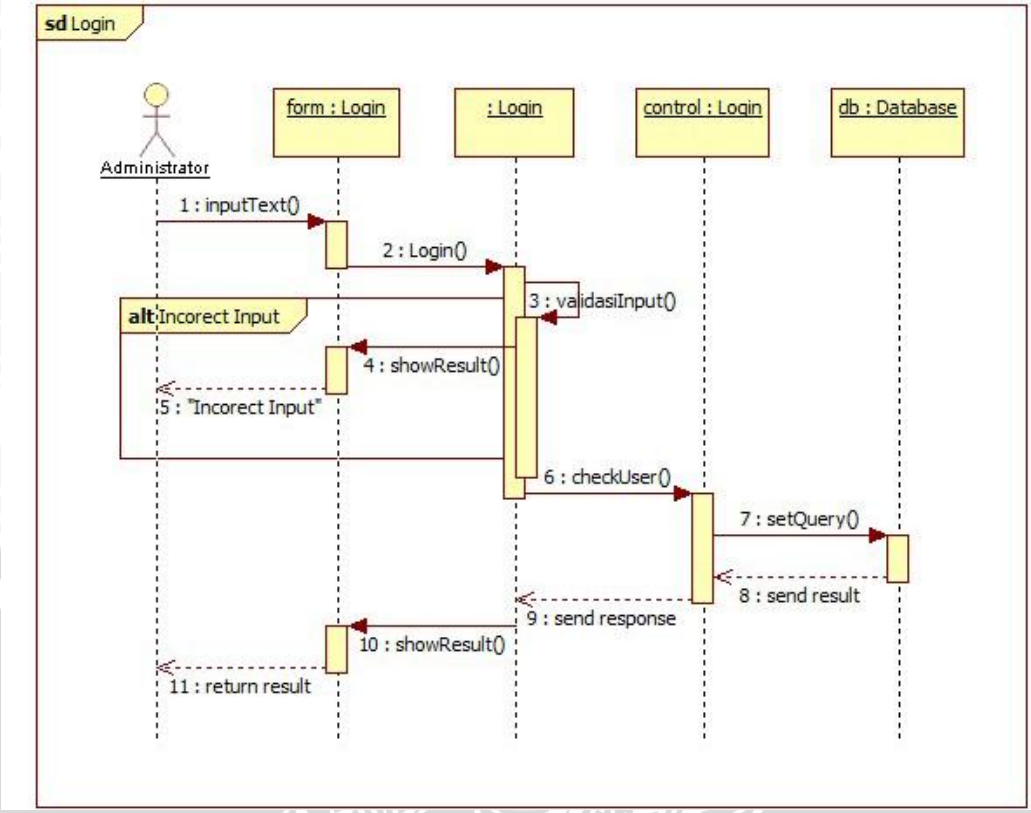
Pada perancangan proses ini ada enam elemen *class* meliputi *class* Gedung, *class* Hubungi, *class* Modul, *class* Node_jarak, *class* Peta, dan *class* User. *Class diagram* memberikan gambaran pemodelan elemen-elemen *class* yang membentuk sebuah sistem perangkat lunak mulai dari atribut-atribut serta fungsi-fungsinya dan relasi dengan *class* lain dalam sebuah sistem. Relasi antar *class* pada perancangan perangkat lunak ini ditunjukan pada Gambar 4.3.

4.2.2 Diagram Sekuensial *(Sequence Diagram)*

Pemodelan aliran jalannya proses interaksi antar objek atau klas yang disusun berdasarkan urutan waktu ditunjukkan oleh *sequence diagram. Sequence diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan keluaran tertentu. Diagram sekuensial (s*equence diagram)* disusun dengan mengambil acuan pada *use case* dan klas yang digambarkan pada *use case* tersebut.

4.2.2.1 *Sequence Diagram* untuk *Login*

Gambar 4.4 merupakan diagram sekuensial pada proses *Login*. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan proses *login* ke dalam sistem perangkat lunak tersebut. Terdapat peringatan jika *field* yang diisikan salah. *Sequence diagram login* adalah sebagai berikut:



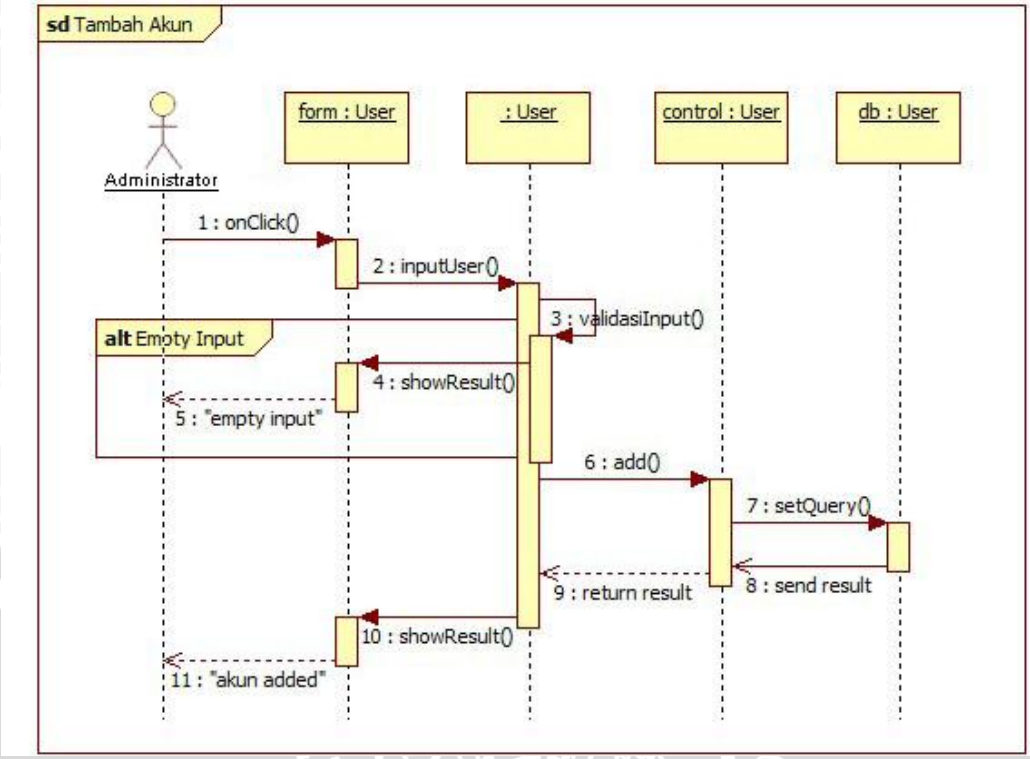
Gambar 4.4 *Sequence Diagram Login*

Sumber: [Perancangan]

Proses *login* dijelaskan dengan aktifitas *administrator* masuk ke halaman *login* dengan memasukkan *username* dan *password.* Sistem akan melakukan validasi (jika sesuai akan terhubung ke *database* dan jika *username* dan *password* yang diisikan salah sistem akan menampilkan pesan *error* ke *user*). Kemudian *user* mengisikan kembali data pada *field* tersebut, jika benar sistem akan menampilkan konfirmasi telah berhasil *login* ke *administrator*.

4.2.2.2 *Sequence Diagram* Tambah Akun

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan tambah akun. Gambar 4.5 merupakan diagram sekuensial untuk melakukan tambah akun. *Sequence diagram* tambah akun adalah sebagai berikut:

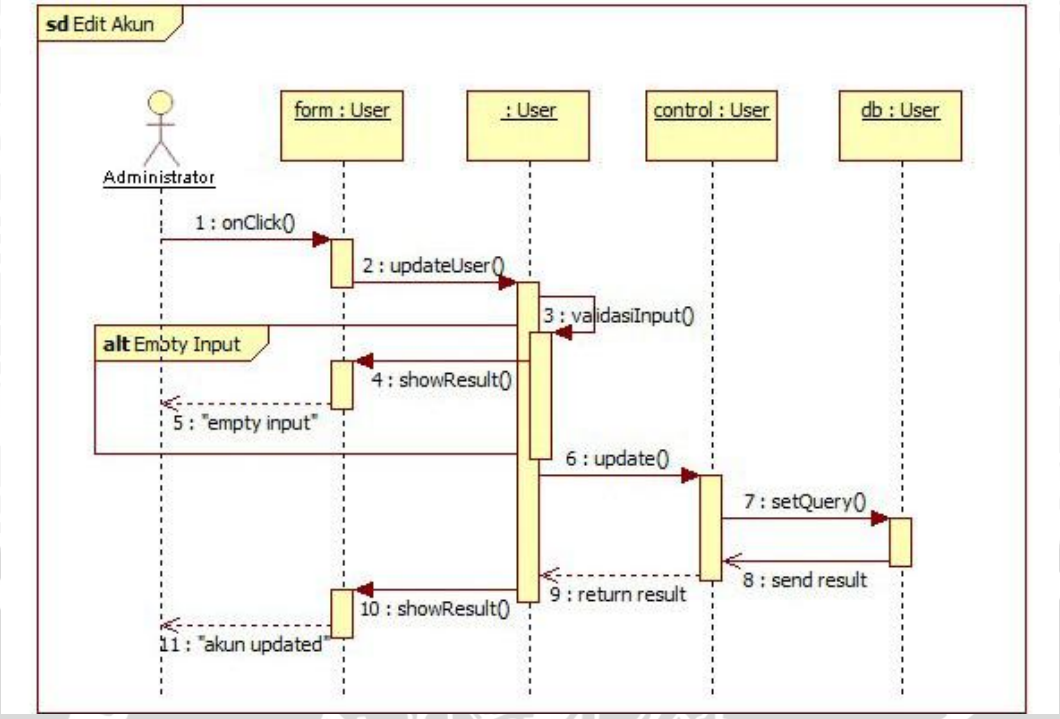


Gambar 4.5 *Sequence Diagram* Tambah Akun
Sumber: [Perancangan]

Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penambahan akun *administrator* baru. Proses dijelaskan dengan *administrator* memilih menu manajemen *user*, dan melakukan pendaftaran akun baru. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan data *administrator* baru disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa pendaftaran berhasil.

4.2.2.3 *Sequence Diagram* Edit Akun

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan edit akun. Gambar 4.6 merupakan diagram sekuensial untuk melakukan edit. Diagram ini menggambarkan interaksi ketika *administrator* melakukan perubahan data akun *administrator* yang ada. Proses edit akun dijelaskan dengan *administrator* memilih menu manajemen *user*, kemudian memilih akun yang akan diedit dengan merubah *username* dan *password*.



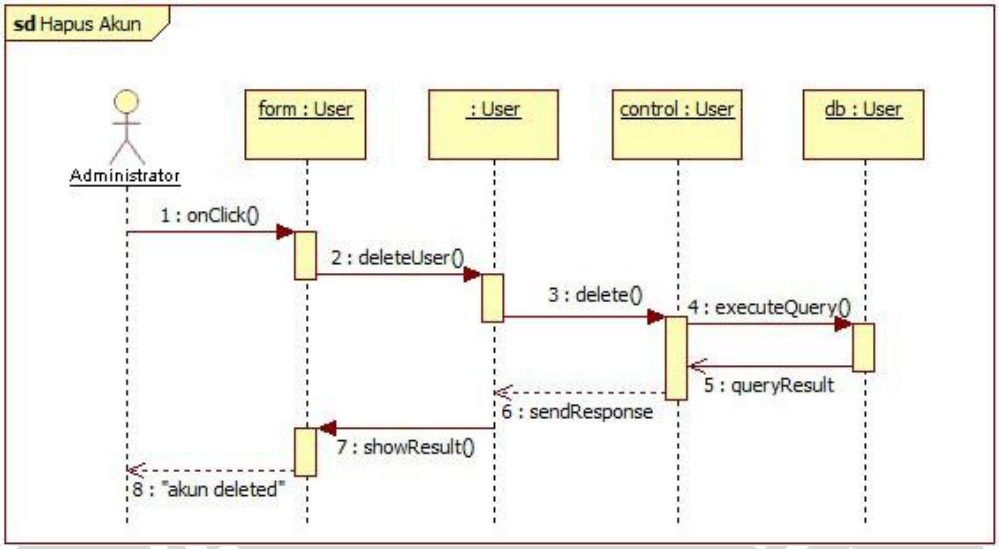
Gambar 4.6 *Sequence Diagram* Edit Akun

Sumber: [Perancangan]

Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan data *administrator* disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa perubahan akun telah berhasil.

4.2.2.4 *Sequence Diagram* Hapus Akun

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus akun. Gambar 4.7 merupakan diagram sekuensial untuk hapus akun. Diagram ini menggambarkan interaksi ketika *administrator* melakukan penghapusan data akun *administrator* yang ada. Proses hapus akun dijelaskan dengan *administrator* memilih menu manajemen *user*, kemudian memilih akun yang dihapus dengan menekan tombol aksi hapus. Jika proses berhasil, sistem akan terhubung ke *database* dan data *administrator* akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan akun telah berhasil. *Sequence diagram* hapus akun adalah sebagai berikut:

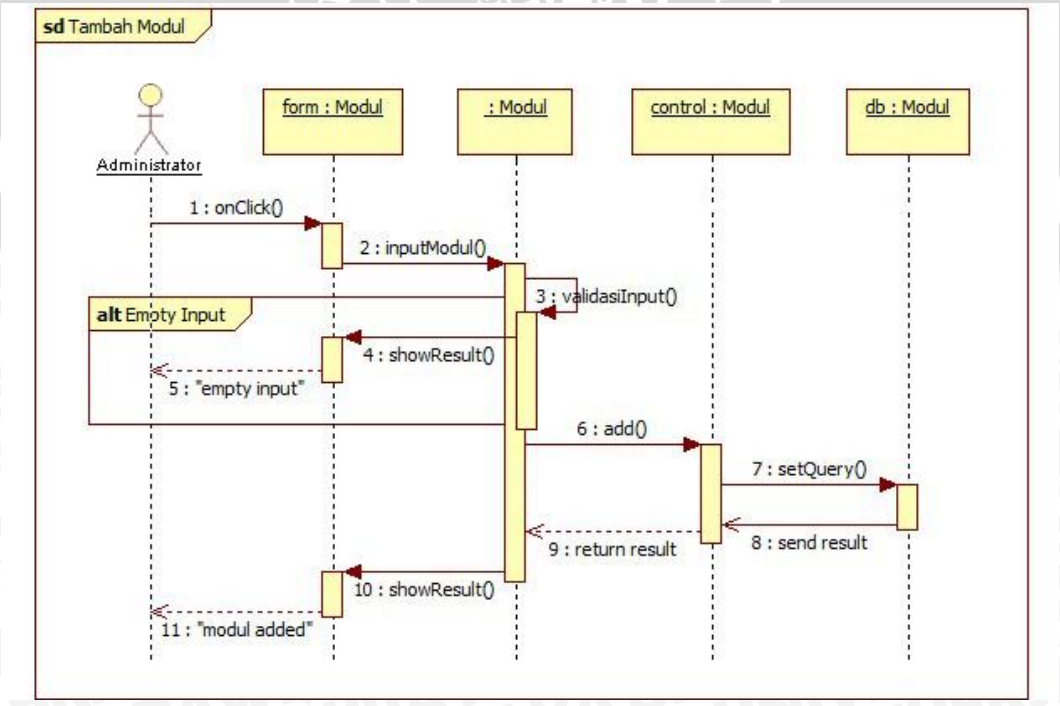


Gambar 4.7 *Sequence Diagram* Hapus Akun

Sumber: [Perancangan]

4.2.2.5 *Sequence Diagram* Tambah Modul

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan tambah modul. Gambar 4.8 merupakan diagram untuk melakukan tambah modul. *Sequence diagram* tambah modul adalah sebagai berikut:



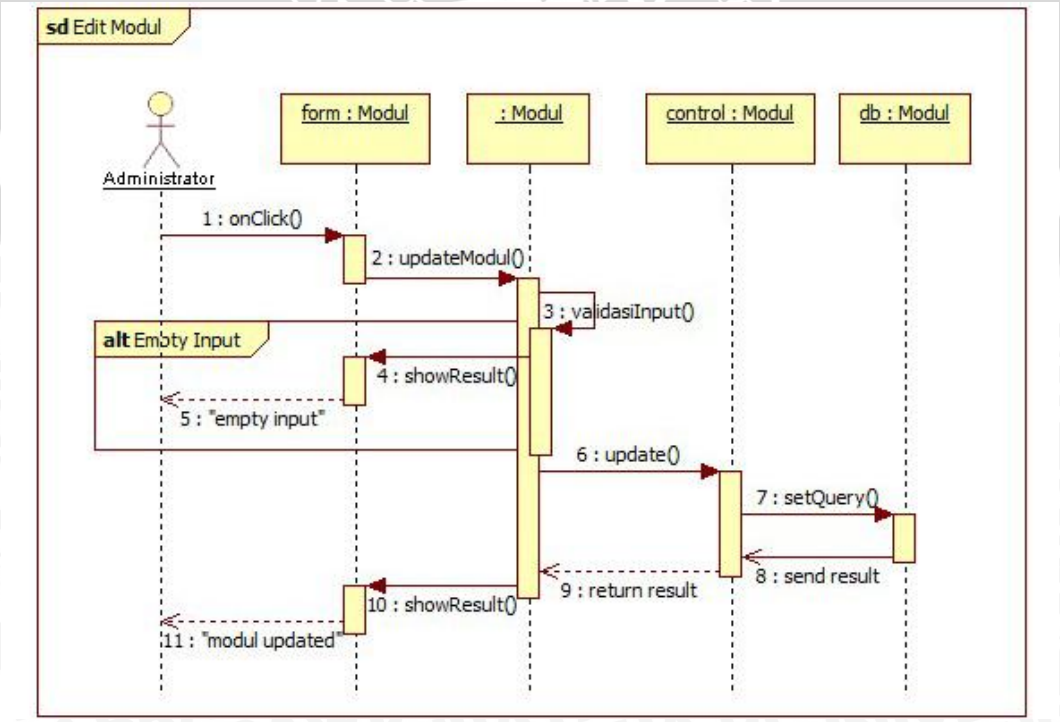
Gambar 4.8 *Sequence Diagram* Tambah Modul

Sumber: [Perancangan]

Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penambahan modul baru. Proses tambah modul dijelaskan dengan *administrator* memilih menu manajemen modul, kemudian mengisi nama modul, link, status, dan urutan modul di form tambah modul. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan modul baru disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa penambahan modul telah berhasil.

4.2.2.6 *Sequence Diagram* Edit Modul

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan edit modul. Gambar 4.9 merupakan diagram sekuensial untuk melakukan edit modul. Diagram ini menggambarkan interaksi saat *administrator* melakukan perubahan modul yang ada. Proses edit modul dijelaskan dengan *administrator* memilih menu manajemen modul, dan memilih modul yang akan diedit dengan merubah nama modul, link, status, dan urutan modul. *Sequence diagram* edit modul adalah sebagai berikut:



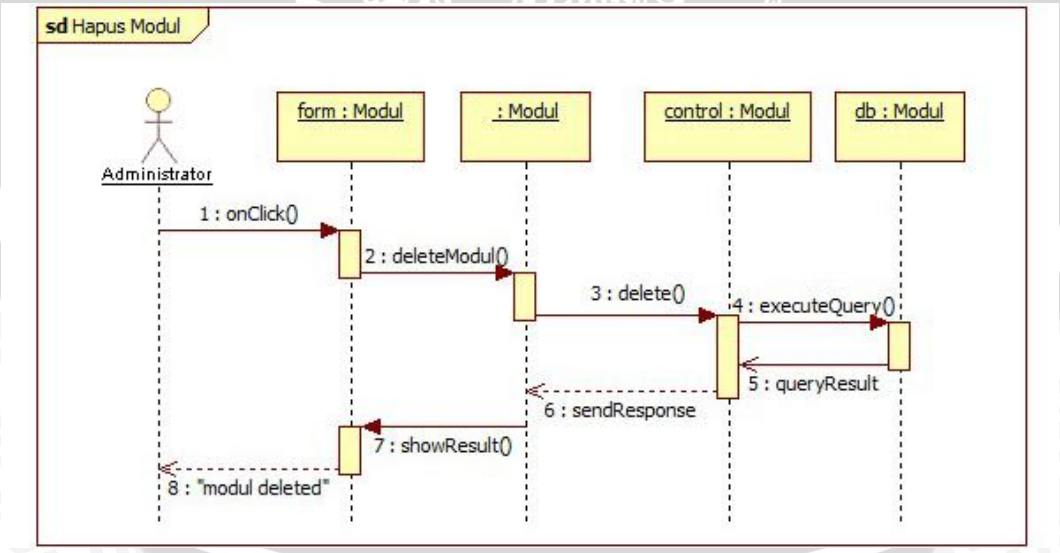
Gambar 4.9 *Sequence Diagram* Edit Modul

Sumber: [Perancangan]

Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan modul disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa perubahan modul telah berhasil.

4.2.2.7 *Sequence Diagram* Hapus Modul

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus modul. Gambar 4.10 merupakan diagram sekuensial untuk melakukan hapus modul. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penghapusan modul yang ada. Proses hapus modul dijelaskan dengan *administrator* memilih menu manajemen modul, kemudian memilih modul yang akan dihapus dengan menekan tombol aksi hapus. Jika proses berhasil, sistem akan terhubung ke *database* dan modul akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan modul telah berhasil. *Sequence diagram* hapus modul adalah sebagai berikut:



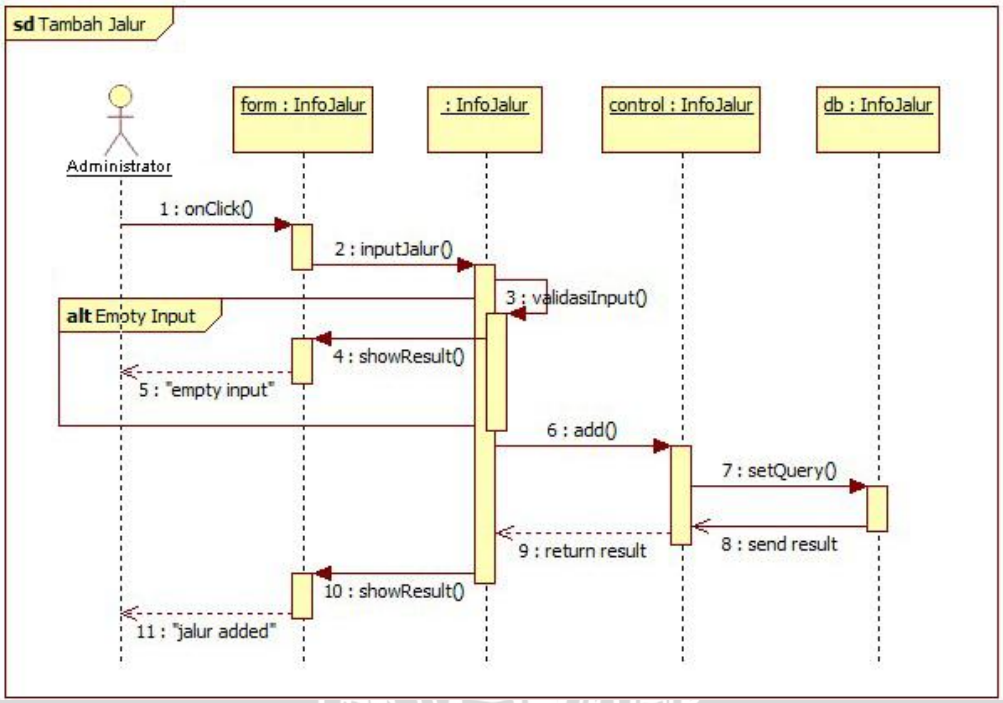
Gambar 4.10 *Sequence Diagram* Hapus Modul

Sumber: [Perancangan]

4.2.2.8 *Sequence Diagram* Tambah Rute

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan tambah rute. Gambar 4.11 merupakan diagram sekuensial untuk

melakukan tambah rute. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penambahan rute baru. Proses tambah rute dijelaskan dengan *administrator* memilih menu jalur, kemudian mengisi *node* awal, *node* akhir, keterangan, dan jarak di form tambah jalur. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan rute baru disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa penambahan rute telah berhasil.

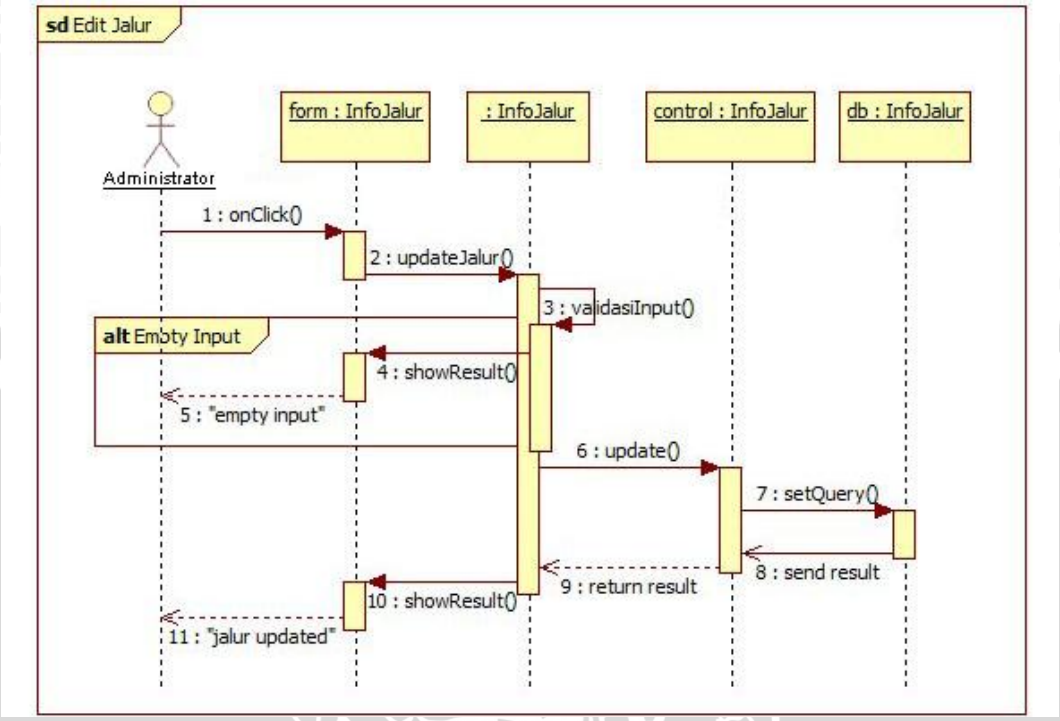


Gambar 4.11 *Sequence Diagram* Tambah Rute
Sumber: [Perancangan]

4.2.2.9 *Sequence Diagram* Edit Rute

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan edit rute. Gambar 4.12 merupakan diagram sekuensial untuk melakukan edit rute. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan perubahan rute yang ada. Proses edit rute dijelaskan dengan *administrator* memilih menu jalur, kemudian memilih rute yang akan diedit dengan merubah *node* awal, *node* akhir, keterangan, dan jarak di form jalur. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan

rute disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa perubahan rute telah berhasil. *Sequence diagram* edit rute adalah sebagai berikut:

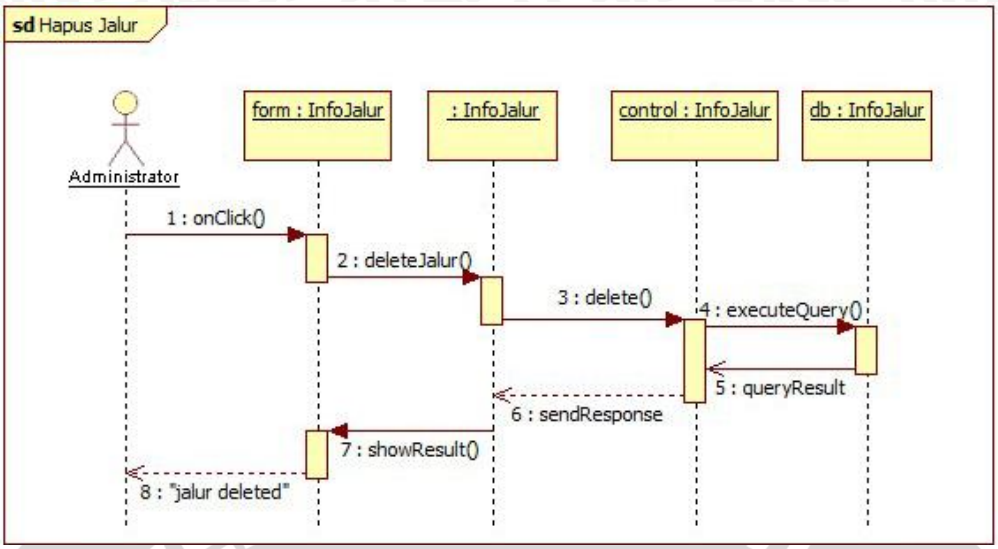


Gambar 4.12 *Sequence Diagram* Edit Rute

Sumber: [Perancangan]

4.2.2.10 *Sequence Diagram* Hapus Rute

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus rute. Gambar 4.13 merupakan diagram sekuensial untuk melakukan hapus rute. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penghapusan rute yang ada. Proses hapus rute dijelaskan dengan *administrator* memilih menu jalur, kemudian memilih rute yang akan dihapus dengan menekan tombol aksi hapus. Jika proses berhasil, sistem akan terhubung ke *database* dan rute akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan rute telah berhasil. Jika proses tidak berhasil maka sistem akan menampilkan pesan *error* kepada *user*. *Sequence diagram* hapus rute adalah sebagai berikut:

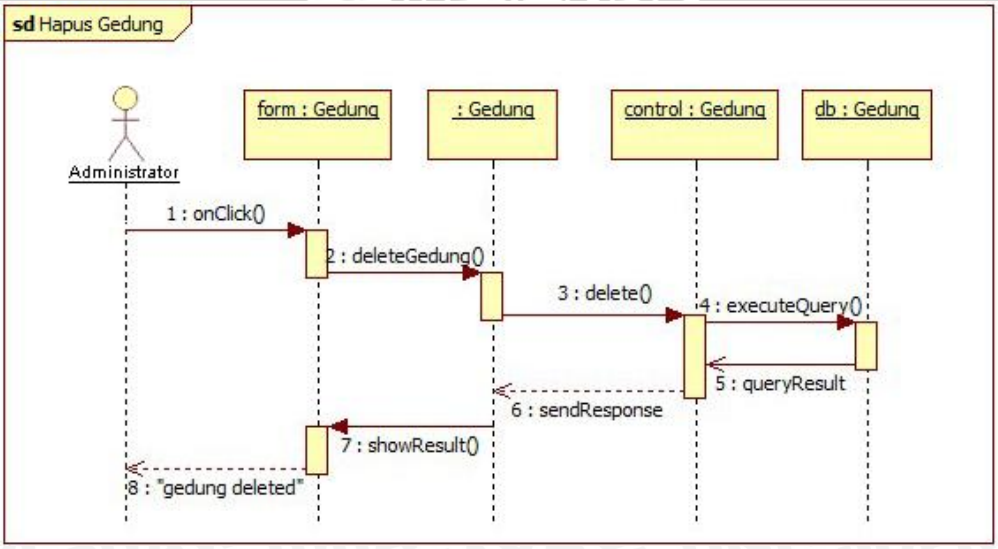


Gambar 4.13 *Sequence Diagram* Hapus Rute

Sumber: [Perancangan]

4.2.2.11 *Sequence Diagram* Hapus Gedung

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus gedung. Gambar 4.14 merupakan diagram sekuensial untuk melakukan hapus gedung. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penghapusan gedung yang ada. Proses hapus gedung dijelaskan dengan *administrator* memilih menu gedung, kemudian memilih gedung yang akan dihapus dengan menekan tombol aksi hapus.



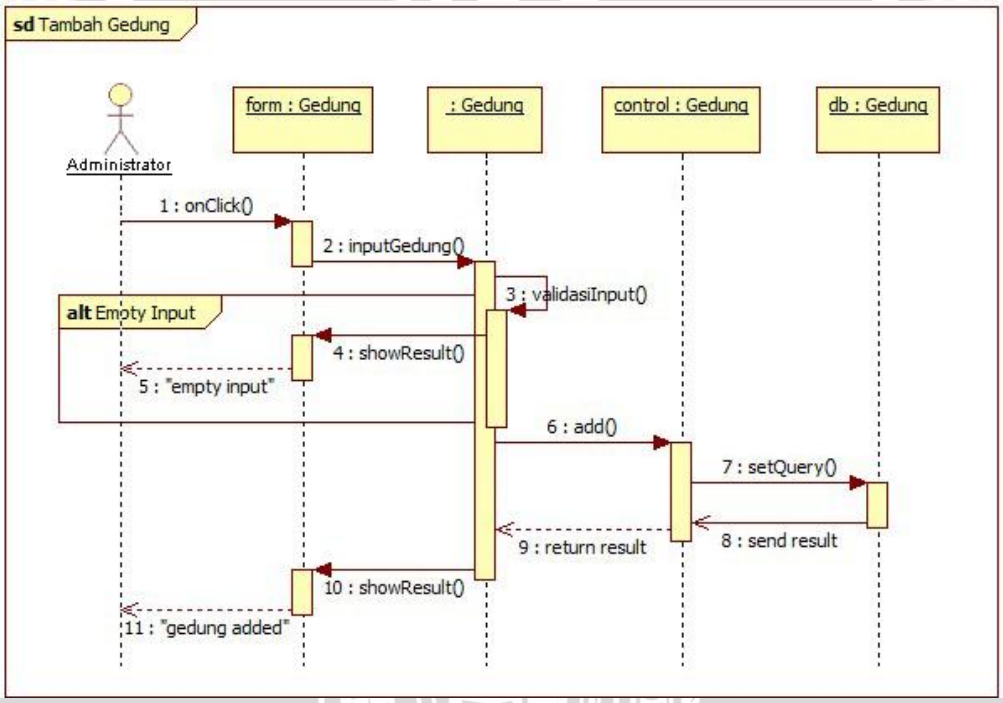
Gambar 4.14 *Sequence Diagram* Hapus Gedung

Sumber: [Perancangan]

Jika proses berhasil, sistem akan terhubung ke *database* dan gedung akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan gedung telah berhasil.

4.2.2.12 *Sequence Diagram* Tambah Gedung

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan tambah gedung. Gambar 4.15 merupakan diagram sekuensial untuk melakukan tambah gedung.

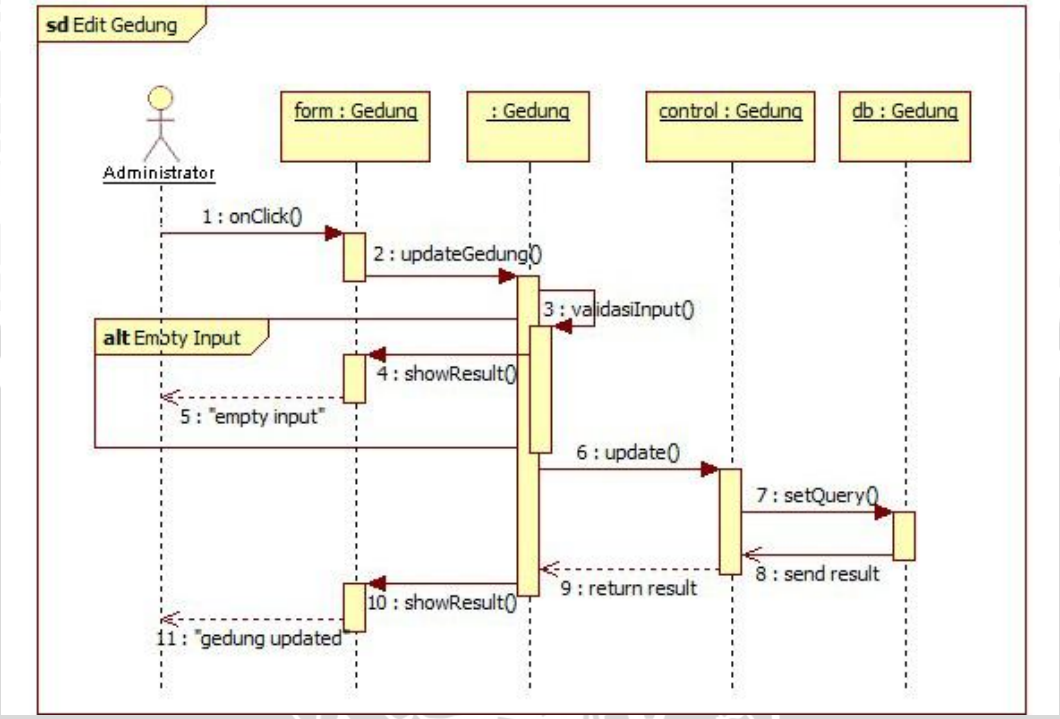


Gambar 4.15 *Sequence Diagram* Tambah Gedung
Sumber: [Perancangan]

Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penambahan gedung baru. Proses tambah gedung dijelaskan dengan *administrator* memilih menu gedung, kemudian mengisi id gedung, dan nama gedung di form gedung. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan gedung baru disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa penambahan gedung telah berhasil.

4.2.2.13 *Sequence Diagram* Edit Gedung

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan edit gedung. Gambar 4.16 merupakan diagram sekuensial untuk melakukan edit gedung.



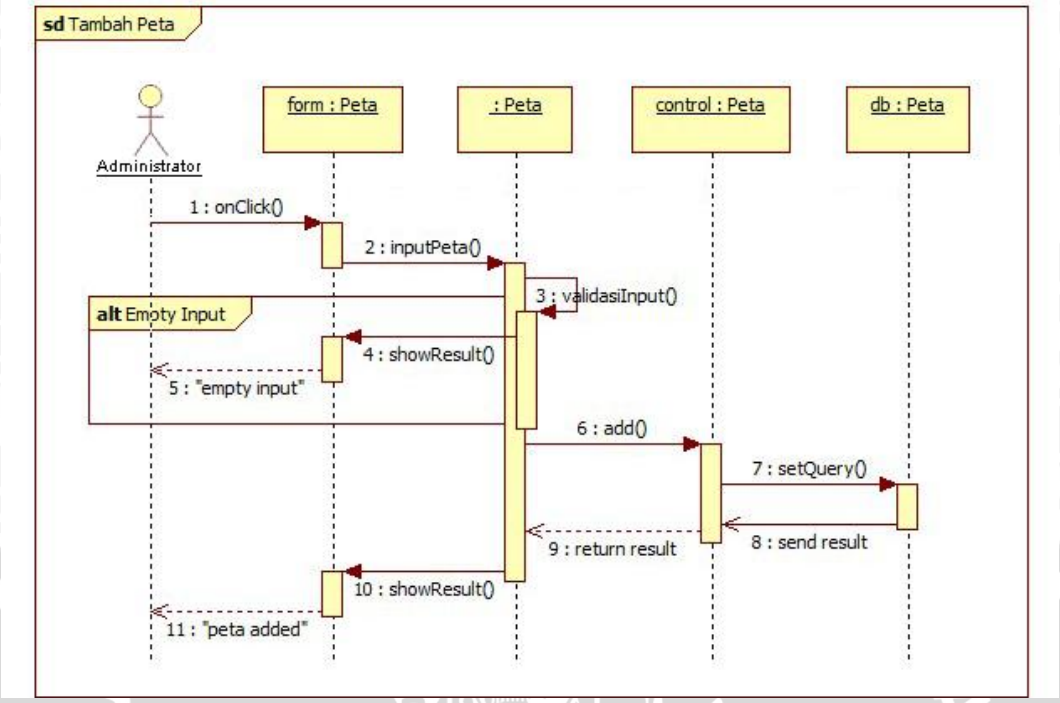
Gambar 4.16 *Sequence Diagram* Edit Gedung

Sumber: [Perancangan]

Diagram ini menggambarkan interaksi *administrator* melakukan edit gedung. Proses edit gedung dijelaskan *administrator* memilih menu gedung, kemudian memilih gedung yang akan diedit dengan merubah id gedung, dan nama gedung di form gedung. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan gedung disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa perubahan gedung telah berhasil.

4.2.2.14 *Sequence Diagram* Tambah Peta

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan tambah peta. Gambar 4.17 merupakan diagram sekuensial untuk melakukan tambah peta.



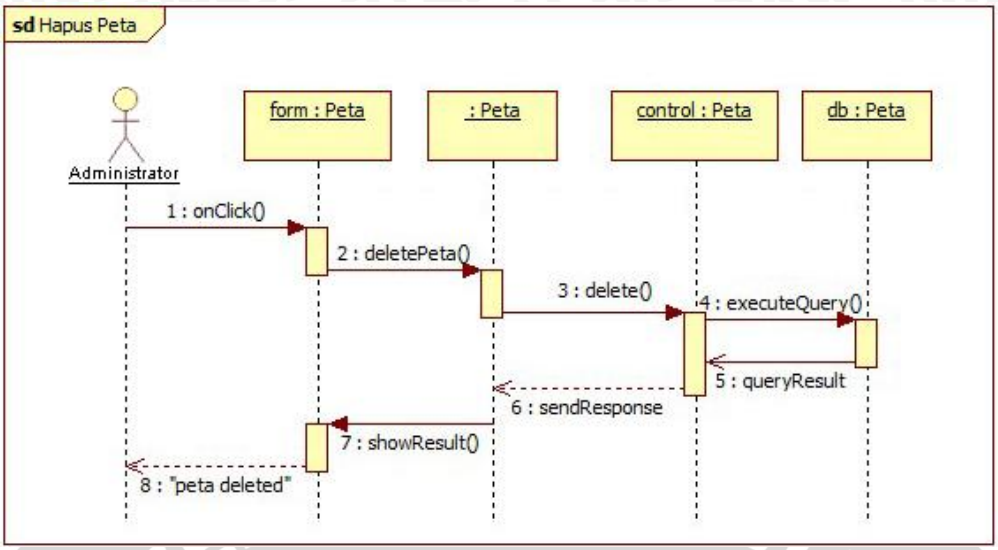
Gambar 4.17 *Sequence Diagram* Tambah Peta

Sumber: [Perancangan]

Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penambahan peta baru. Proses tambah peta dijelaskan dengan *administrator* memilih menu peta, kemudian mengisi *node* awal, *node* akhir, *browse* file, dan keterangan di form peta. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan peta baru disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa penambahan peta telah berhasil.

4.2.2.15 *Sequence Diagram* Hapus Peta

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus peta. Gambar 4.18 merupakan diagram sekuensial untuk melakukan hapus peta. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penghapusan peta yang ada. *Sequence diagram* hapus peta adalah sebagai berikut:



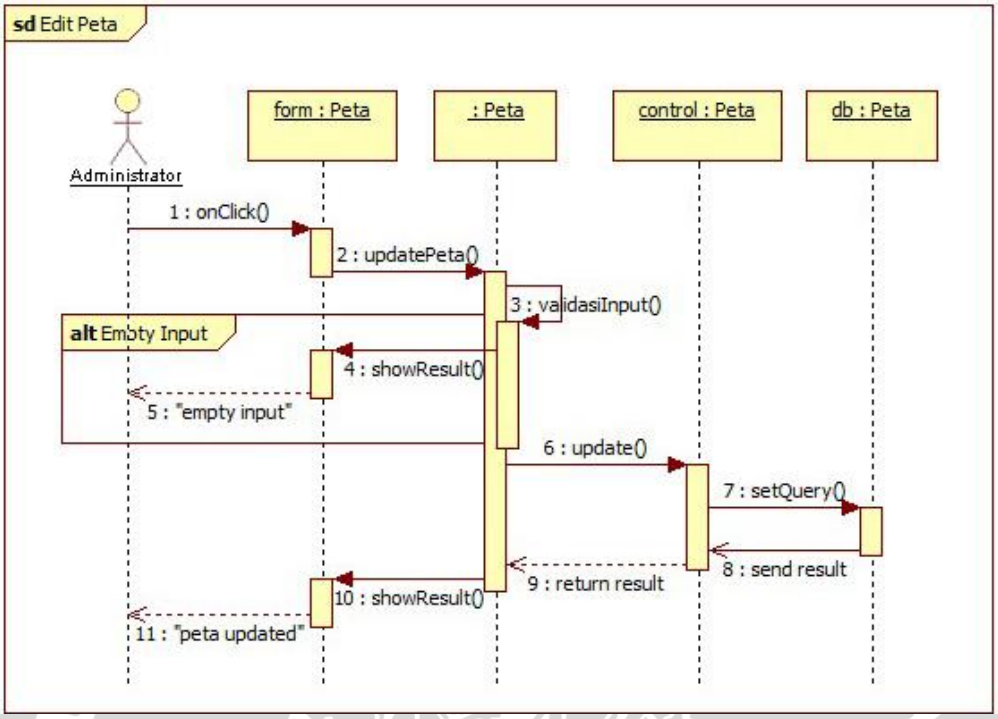
Gambar 4.18 *Sequence Diagram* Hapus Peta

Sumber: [Perancangan]

Proses hapus peta dijelaskan dengan *administrator* memilih menu peta, kemudian memilih peta yang akan dihapus dengan menekan tombol aksi hapus. Jika proses berhasil, sistem akan terhubung ke *database* dan peta akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan peta telah berhasil.

4.2.2.16 *Sequence Diagram* Edit Peta

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan edit peta. Gambar 4.19 merupakan diagram sekuensial untuk melakukan edit peta. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan perubahan peta yang ada. Proses edit peta dijelaskan dengan *administrator* memilih menu peta, kemudian memilih peta yang akan diedit dengan merubah *node* awal, *node* akhir, *browse* file, dan keterangan di form peta. Jika data yang diisikan sesuai, sistem akan terhubung ke *database* dan peta disimpan di *database* (jika data tidak sesuai sistem akan menampilkan pesan *error*). Sistem akan memberikan konfirmasi ke *administrator* bahwa perubahan peta telah berhasil. *Sequence diagram* edit peta adalah sebagai berikut:

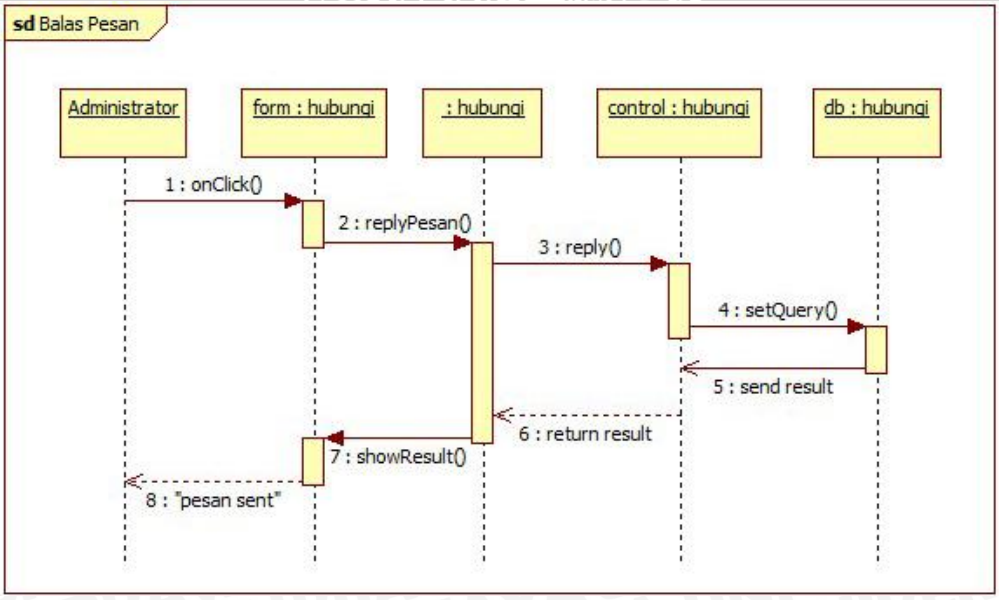


Gambar 4.19 *Sequence Diagram* Edit Peta

Sumber: [Perancangan]

4.2.2.17 *Sequence Diagram* Balas Pesan

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan balas pesan. *Sequence diagram* balas pesan adalah sebagai berikut:



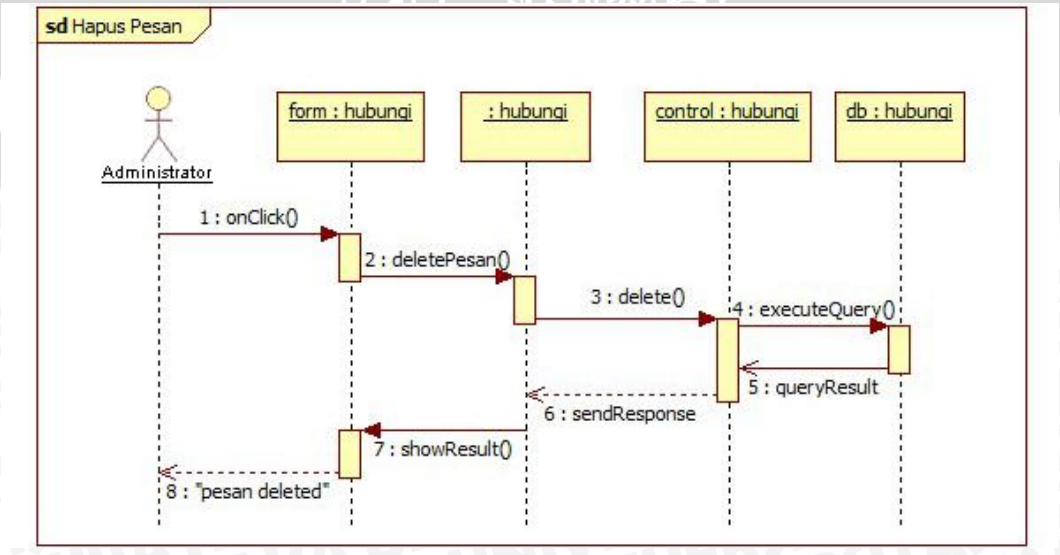
Gambar 4.20 *Sequence Diagram* Balas Pesan

Sumber: [Perancangan]

Gambar 4.20 merupakan diagram sekuensial untuk melakukan balas pesan. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan balas pesan yang dikirim oleh *user*. Proses balas pesan dijelaskan dengan *administrator* memilih menu hubungi, kemudian memilih pesan yang akan dibalas. Jika data yang diisikan sesuai, pesan akan terkirim kepada email *user*. Sistem akan memberikan konfirmasi ke *administrator* bahwa pengiriman pesan telah berhasil.

4.2.2.18 *Sequence Diagram* Hapus Pesan

Sequence diagram selanjutnya adalah *sequence diagram* untuk melakukan hapus pesan. Gambar 4.21 merupakan diagram sekuensial untuk melakukan hapus pesan. Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan penghapusan pesan yang ada. Proses hapus pesan dijelaskan dengan *administrator* memilih menu hubungi, kemudian memilih pesan yang akan dihapus dengan menekan tombol aksi hapus. Jika proses berhasil, sistem akan terhubung ke *database* dan pesan akan terhapus. Sistem akan memberikan konfirmasi ke *administrator* bahwa penghapusan pesan telah berhasil. *Sequence diagram* hapus pesan adalah sebagai berikut:

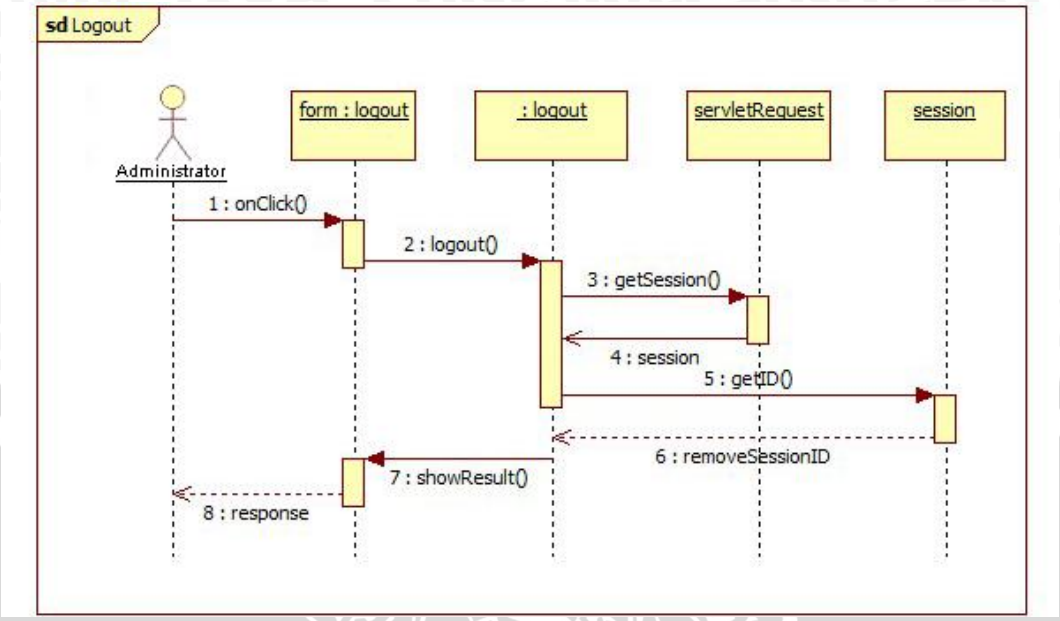


Gambar 4.21 *Sequence Diagram* Hapus Pesan

Sumber: [Perancangan]

4.2.2.19 *Sequence Diagram Logout*

Sequence diagram selanjutnya adalah *sequence diagram* untuk keluar dari sistem. Gambar 4.22 merupakan diagram sekuensial untuk *logout*.



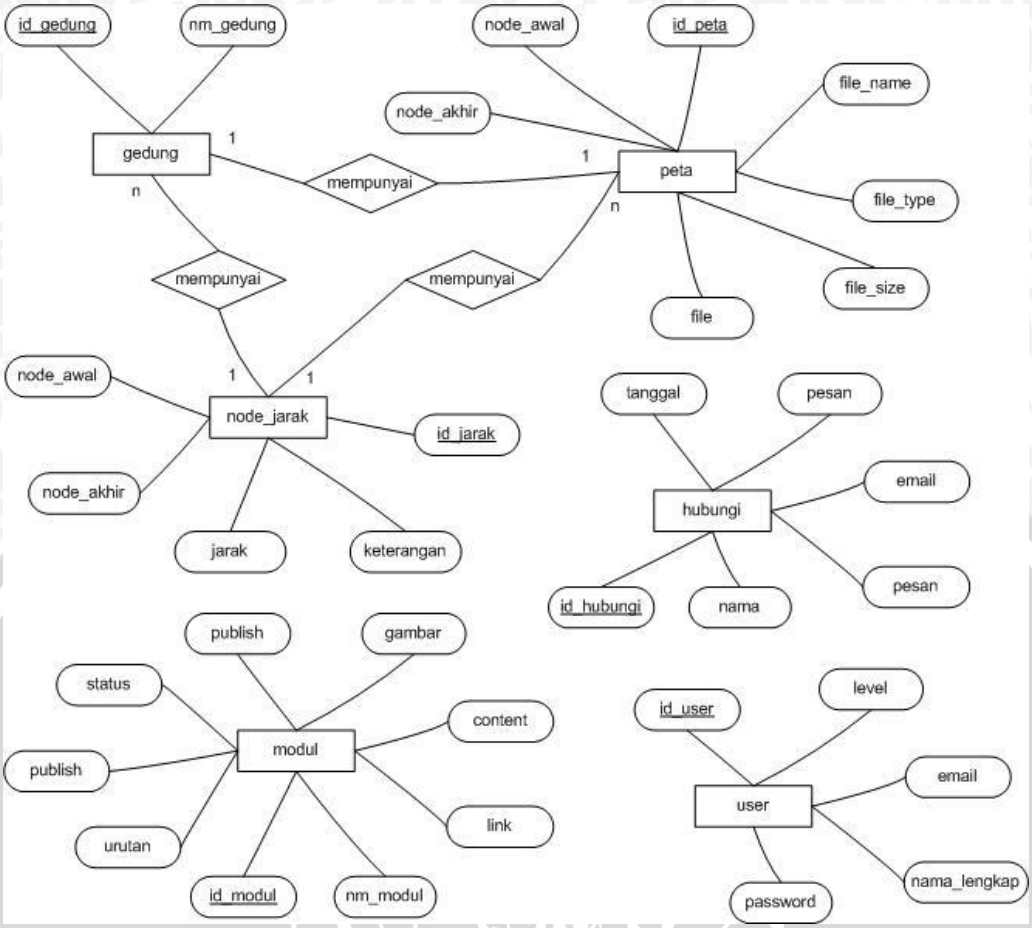
Gambar 4.22 *Sequence Diagram Logout*

Sumber: [Perancangan]

Diagram sekuensial ini menggambarkan interaksi ketika *administrator* melakukan proses *logout*. Proses *logout* dijelaskan dengan *administrator* memilih menu logout, kemudian sistem akan terhubung ke *database* dan *session* berakhir. Sistem akan memberikan konfirmasi ke *administrator* bahwa akun telah berhasil keluar dari sistem.

4.2.3 Perancangan Basis Data

Basis data berfungsi sebagai tempat menyimpan data. Pada skripsi ini perancangan basis data direpresentasikan dalam bentuk *Entity Relationship Diagram* (ERD). ERD sistem ini dapat dilihat pada Gambar 4.23. ERD menunjukkan hubungan yang terjadi diantara objek (entitas) yang terlibat dalam suatu *database*. ERD berisi komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan beberapa atribut yang mempresentasikan seluruh fakta yang ditinjau dari keadaan yang nyata.



Gambar 4.23 ERD aplikasi web pencarian rute terpendek

Sumber: [Perancangan]

Pada perancangan basis data sistem ini terdapat enam buah tabel yaitu tabel gedung, tabel hubungi, tabel modul, tabel node_jarak, tabel peta, dan tabel *user*.

Berikut ini merupakan struktur tabel serta keterangan masing masing tabel dan *field* yang ada pada *database*. Entitas *user* merepresentasikan tabel *user* yang berisi data-data dari *administrator* seperti *id_user*, nama_lengkap, *password*, level, dan email. Dalam pengembangannya nanti, atribut dari *user* dapat ditambahkan sesuai dengan kebutuhan. Struktur tabel *user* ditunjukkan pada Tabel 4.27.

Tabel 4.27 Struktur tabel *user*

No	Nama Field	Tipe	Lebar	Keterangan
1	id_*user* (PK)	Varchar	50	Kode dari *admin*

2	*password*	Varchar	50	*Password* dari *admin*
3	nama_lengkap	Varchar	100	Nama dari *admin*
4	email	Varchar	100	Email dari *admin*
5	level	Varchar	50	Hak akses *admin*

Sumber: [Perancangan]

Entitas gedung merepresentasikan tabel gedung yang berisi data-data dari gedung-gedung jurusan atau fakultas yang terdapat di kampus UB seperti id_gedung, dan nm_gedung. Dalam pengembangannya nanti, atribut dari gedung dapat ditambahkan sesuai dengan kebutuhan. Struktur tabel gedung ditunjukkan pada Tabel 4.28.

Tabel 4.28 Struktur tabel gedung

No	Nama Field	Tipe	Lebar	Keterangan
1	id_gedung (PK)	Integer	20	Kode dari gedung
2	nm_gedung	Varchar	50	Nama dari gedung

Sumber: [Perancangan]

Entitas hubungi merepresentasikan tabel hubungi yang berisi data-data dari pengguna aplikasi yang mengirimkan pesan kepada *administrator* seperti id_hubungi, nama, email, subjek, pesan, dan tanggal. Dalam pengembangannya nanti, atribut dari hubungi dapat ditambahkan sesuai dengan kebutuhan. Struktur tabel hubungi ditunjukkan pada Tabel 4.29.

Tabel 4.29 Struktur tabel hubungi

No	Nama Field	Tipe	Lebar	Keterangan
1	id_hubungi (PK)	Integer	10	Kode dari hubungi
2	Nama	Varchar	50	Nama dari *user*
3	email	Varchar	100	Email dari *user*
4	subjek	Varchar	100	Subjek pesan dari *user*
5	pesan	Text		Isi pesan dari *user*
6	Tanggal	Date		Tanggal pengiriman *user*

Sumber: [Perancangan]

Entitas modul merepresentasikan tabel modul yang berisi data-data dari modul-modul yang terdapat di dalam aplikasi web pencarian rute terpendek seperti id_modul, nm_modul, *link*, gambar, publish, aktif, dan urutan. Dalam pengembangannya nanti, atribut dari modul dapat ditambahkan sesuai dengan kebutuhan. Struktur tabel modul ditunjukkan pada Tabel 4.30.

Tabel 4.30 Struktur tabel modul

No	Nama Field	Tipe	Lebar	Keterangan
1	id_modul (PK)	Integer	10	Kode dari modul
2	nm_modul	Varchar	50	Nama dari modul
3	*link*	Varchar	100	*Link* dari modul
4	Gambar	Varchar	100	Gambar dari modul
5	Publish	Enum	"Y", "N"	Keterangan dari modul
6	urutan	Integer	5	Urutan dari modul

Sumber: [Perancangan]

Entitas node_jarak merepresentasikan tabel node_jarak yang berisi data-data dari jarak-jarak antar gedung yang terdapat di kampus UB seperti id_jarak, node_awal, node_akhir, keterangan, dan jarak. Dalam pengembangannya nanti, atribut dari node_jarak dapat ditambahkan sesuai dengan kebutuhan. Struktur tabel node_jarak ditunjukkan pada Tabel 4.31.

Tabel 4.31 Struktur tabel node_jarak

No	Nama Field	Tipe	Lebar	Keterangan
1	id_jarak (PK)	Integer	15	Kode dari jarak
2	node_awal	Varchar	25	Titik awal gedung
3	node_akhir	Varchar	25	Titik akhir gedung
4	keterangan	Varchar	100	Keterangan dari jarak
5	Jarak	Varchar	20	Jarak antar gedung

Sumber: [Perancangan]

Entitas peta merepresentasikan tabel peta yang berisi data-data dari peta animasi aplikasi web pencarian rute terpendek seperti id_gedung, dan nm_gedung. Dalam pengembangannya nanti, atribut dari grdung dapat

ditambahkan sesuai dengan kebutuhan. Struktur tabel gedung ditunjukkan pada Tabel 4.32.

Tabel 4.32 Struktur tabel peta

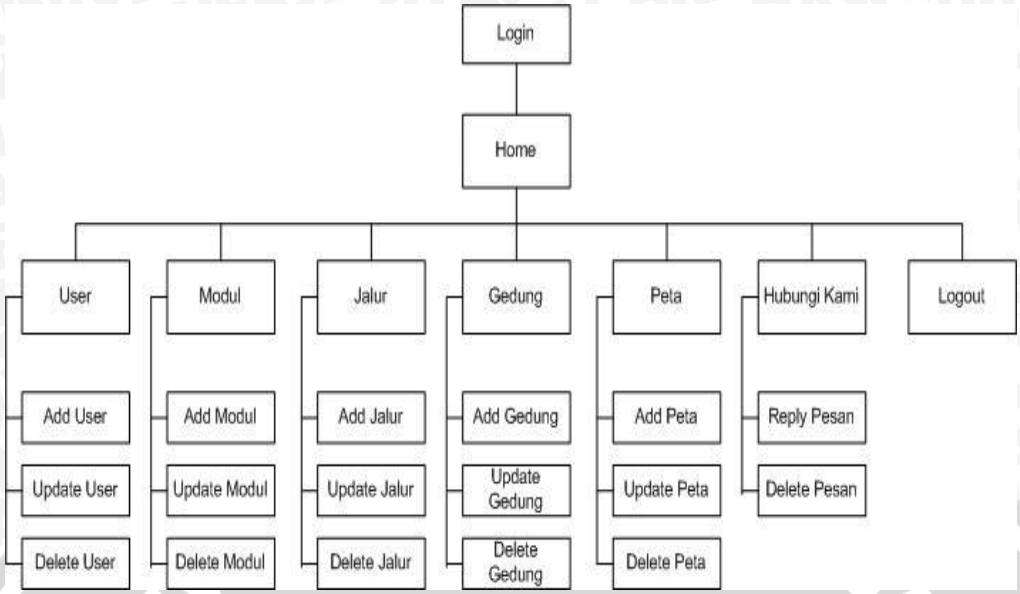
No	Nama Field	Tipe	Lebar	Keterangan
1	id_peta (PK)	Integer	15	Kode dari peta
2	node_awal	Varchar	25	Titik awal gedung
3	node_akhir	Varchar	25	Titik akhir gedung
4	*File*	Blob		*File* yang di-*upload*
5	*file_name*	Varchar	50	Nama dari *file*
6	*file_size*	Integer	15	Ukuran dari *file*
7	*file_type*	Varchar	50	Jenis dari *file*
8	keterangan	Varchar	100	Keterangan dari peta

Sumber: [Perancangan]

4.2.4. Perancangan Antarmuka

Layout aplikasi merupakan gambaran struktur program yang dijalankan. Perancangan antarmuka sangat diperlukan untuk mempermudah *user* dalam menggunakan aplikasi. *Sitemap* digunakan untuk merepresentasikan keseluruhan antarmuka sistem berdasarkan otoritas pengguna. Aplikasi ini mempunyai dua menu otoritas yaitu menu untuk *administrator* dan menu untuk *user*.

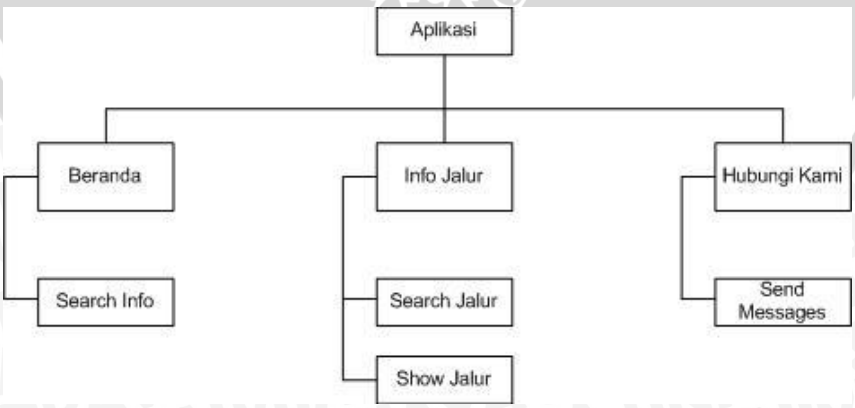
Menu admin merupakan menu khusus yang disediakan oleh aplikasi untuk *administrator*. *Form login* adalah antarmuka pertama yang akan ditampilkan oleh aplikasi. Jika berhasil *login* sebagai *administrator*, maka antarmuka khusus *administrator* yang disediakan oleh aplikasi berupa modul yang berisi menu hak akses yang dapat dilakukan oleh *administrator* seperti manajemen *user*, manajemen modul, manajemen jalur, manajemen gedung, manajemen peta, dan hubungi kami. Di dalam tiap-tiap modul terdapat aksi dari *administrator* yaitu dapat menambah, meng-edit, dan menghapus. Gambar 4.24 menunjukkan *site map* dari menu *adminnistrator*.



Gambar 4.24 *Sitemap* Untuk *Administrator*

Sumber: [Perancangan]

Site map selanjutnya adalah *site map* untuk menu khusus *user*. Menu khusus *user* merupakan antarmuka yang disediakan oleh aplikasi untuk *user*. Halaman Beranda (Home) adalah antarmuka pertama yang akan ditampilkan oleh aplikasi. Pada *home* terdapat fasilitas untuk melihat informasi jurusan atau fakultas yang terdapat di UB. Menu Info Jalur menyediakan fasilitas untuk mencari rute terpendek yang dapat dilalui untuk mencari suatu jurusan atau fakultas yang dibutuhkan. Menu Hubungi Kami merupakan fasilitas untuk *user* dapat mengirimkan pesan berupa pertanyaan, kritik, maupun saran kepada *administrator*.

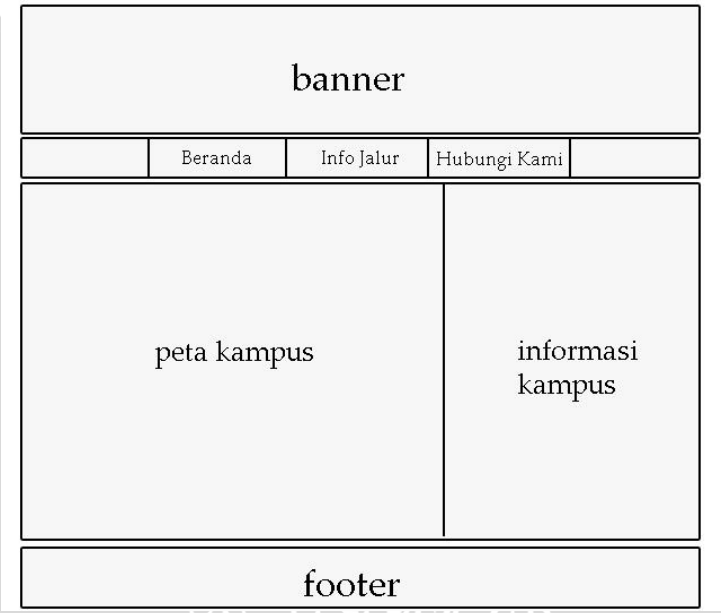


Gambar 4.25 *Sitemap* Untuk *User*

Sumber: [Perancangan]

4.2.4.1 Tampilan Menu Utama (Beranda)

Halaman ini berisi menu utama dari web. *User* dapat memilih dan melihat informasi jurusan atau fakultas yang dibutuhkan, dengan cara mengarahkan kursor dan meng-klik salah satu gambar gedung. Kemudian sistem akan menampilkan informasi berupa data-data yang terdapat pada gedung jurusan atau fakultas. Terdapat gambar gedung jurusan atau fakultas yang dapat memudahkan *user* dalam melakukan pencarian gedung yang dibutuhkannya. Jika user *ingin* mendapatkan informasi yang lebih lengkap mengenai gedung jurusan atau fakultas tersebut, disediakan pula *link* yang langsung mengarahkan ke *website* Universitas Brawijaya. Tampilan menu utama (beranda) ditunjukkan pada Gambar 4.26.

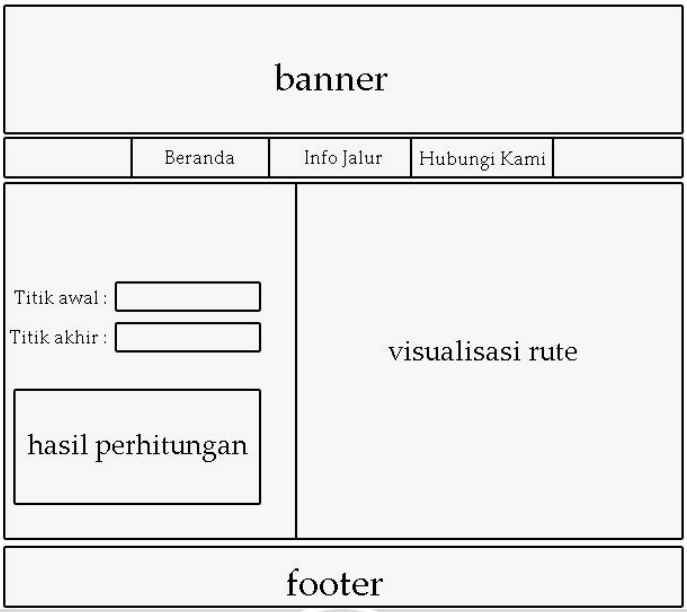


Gambar 4.26 Tampilan Menu Utama (Beranda)

Sumber: [Perancangan]

4.2.4.2 Tampilan Menu Info Jalur

Halaman kedua dari aplikasi web pencarian rute terpendek antar jurusan ini berisi menu info jalur. Menu ini memungkinkan *user* dapat mencari rute terpendek yang akan dilalui untuk menuju sebuah jurusan atau fakultas yang dibutuhkan. Prosesnya yaitu *user* memilih titik awal dan titik akhir dari gedung jurusan atau fakultas yang dibutuhkan. Tampilan menu info jalur ditunjukkan pada Gambar 4.27



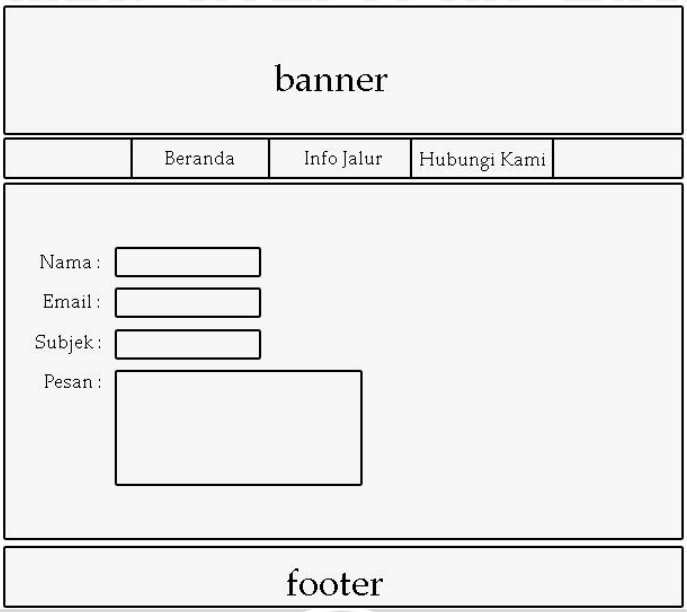
Gambar 4.27 Tampilan Menu Info Jalur

Sumber: [Perancangan]

Hasil perhitungan pencarian rute terpendek menggunakan algoritma Floyd-warshall. Kemudian sistem akan menampilkan hasil perhitungan rute dengan menampilkan panjang jarak yang dapat dilalui serta rute gedung mana saja yang dapat dilewati. Hasil rute tersebut akan divisualisasikan melalui animasi peta kampus Universitas Brawijaya dua dimensi berformat *Adobe Flash* (**.swf*).

4.2.4.3 Tampilan Menu Hubungi Kami

Halaman ini berisi menu hubungi kami. *User* dapat menulis pesan berupa saran, kritik, maupun pertanyaan kepada *administrator*. *User* terlebih dahulu mengisikan data-data pada *field* yang berupa nama, *email*, subjek, dan isi pesan. Setelah *user* mengisikan data tersebut maka akan keluar pemberitahuan bahwa pesan telah berhasil dikirim kepada *administrator*. Jika terdapat salah satu *field* yang masih kosong maka sistem akan menampilkan peringatan kepada *user* bahwa semua *field* harus diisi. Tampilan menu info jalur ditunjukkan pada Gambar 4.28

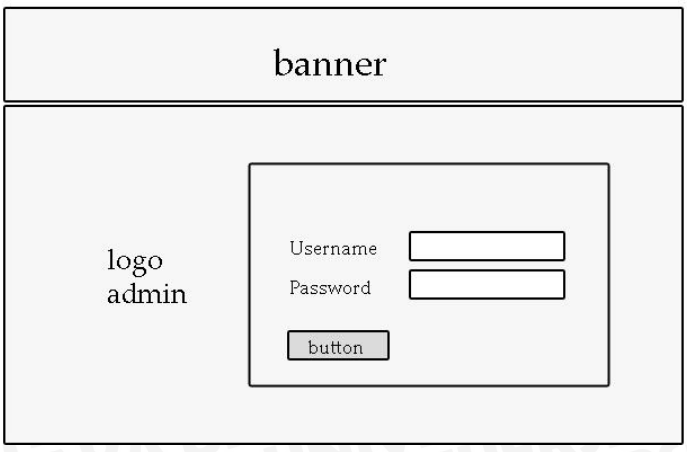


Gambar 4.28 Tampilan Menu Hubungi Kami

Sumber: [Perancangan]

4.2.4.4 Tampilan Menu Halaman *Login*

Halaman ini berisi menu *login* untuk *administrator*. *Administrator* dapat mengakses menu ini dengan memasukkan *username* dan *password* terlebih dahulu. Jika salah satu *field* masih kosong maka akan muncul peringatan bahwa *administrator* harus mengisi semia *field* yang ada. Tampilan menu *login administrator* ditunjukkan pada Gambar 4.29

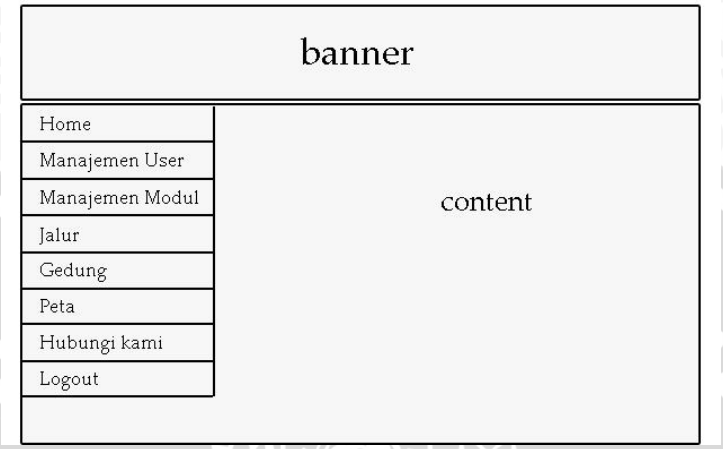


Gambar 4.29 Tampilan Menu *Login Administrator*

Sumber: [Perancangan]

4.2.4.5 Tampilan Menu Halaman *Administrator*

Halaman ini berisi menu *administrator* yang terdiri dari berbagai modul yaitu, manajemen *user*, manajemen modul, jalur, gedung, peta, dan hubungi kami. Tampilan menu *administrator* ditunjukkan pada Gambar 4.30.

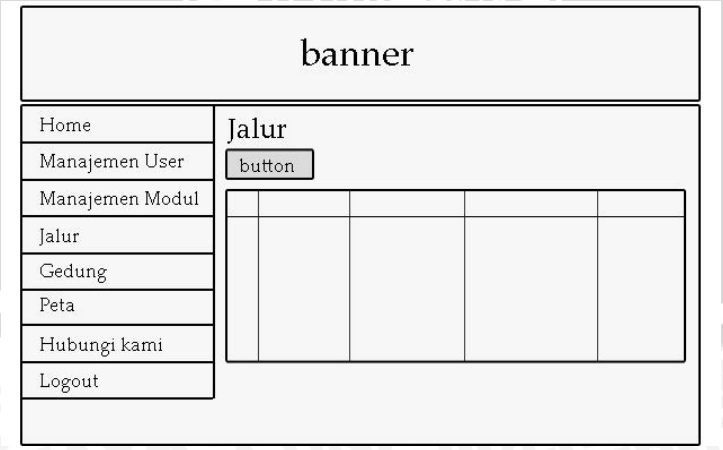


Gambar 4.30 Tampilan Menu *Administrator*

Sumber: [Perancangan]

4.2.4.6 Tampilan Menu Manajemen Jalur

Halaman ini berisi menu untuk menampilkan semua rute yang ada di kampus UB, juga terdapat aksi tambah, hapus dan edit rute yang dapat dilakukan oleh *admin*. Tampilan menu manajemen jalur ditunjukkan pada Gambar 4.31.

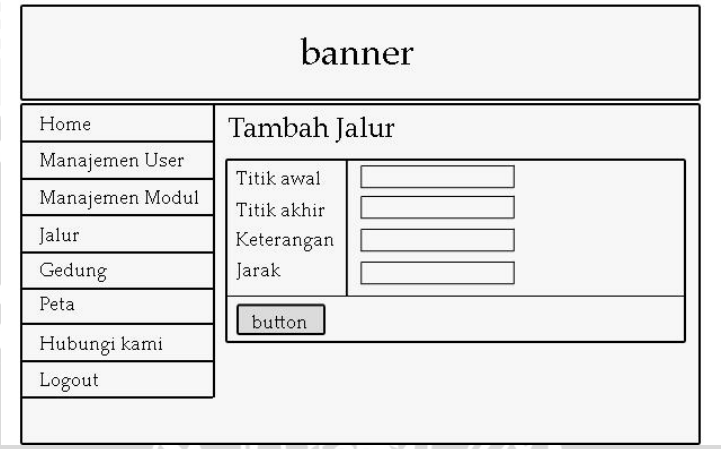


Gambar 4.31 Tampilan Menu Manajemen Jalur

Sumber: [Perancangan]

4.2.4.7 Tampilan Menu Tambah Jalur

Halaman ini berisi menu untuk menambahkan jalur baru yang terdapat di kampus, *administrator* memasukkan titik awal, titik akhir, keterangan, dan jarak pada menu ini. Tampilan menu manajemen jalur ditunjukkan pada Gambar 4.32.



Gambar 4.32 Tampilan Menu Tambah Jalur

Sumber: [Perancangan]