

IV. HASIL DAN PEMBAHASAN

4.1 Gambaran umum perusahaan

Gambaran umum perusahaan merupakan penggambaran dari perusahaan yang menjelaskan tentang sejarah umum, visi, misi dan struktur organisasi *Software House* Lampung.

4.1.1 Sejarah umum perusahaan

Software House Lampung adalah salah satu perusahaan yang bergerak di bidang solusi teknologi informasi yang didirikan pada 5 September 2015. *Software House* Lampung beralamatkan di Jalan Sultan Haji No. 48 Kota Sepang, Bandar Lampung. *Software House* Lampung siap melayani berbagai kebutuhan personal maupun bisnis, dan juga melayani jasa pembuatan *website company profile*, portal berita *online*, toko *online*, *custom*, *software* bisnis, *software* akunting, jasa *Internet Marketing*, *hosting* dan *domain*.

4.1.2 Visi perusahaan

Visi dari *Software House* Lampung adalah ingin menjadi perusahaan yang paling inovatif di Lampung yang memperkaya kehidupan pribadi pelanggan dan membuat bisnis lebih berhasil dengan membawa ke pasar kreatif dan menghasilkan iklan berorientasi dan layanan komunikasi, serta membangun tim dan *sharing owner*.

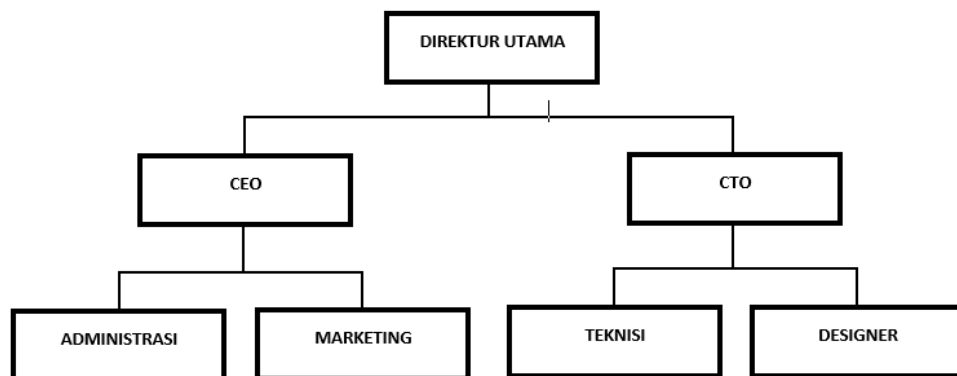
4.1.3 Misi perusahaan

Misi dari *Software House* Lampung adalah :

1. Menjadi sahabat dan konsultan untuk pelanggan dalam bisnis industri masing - masing.
2. Memberikan keunggulan operasional untuk pelanggan dalam bisnis industri masing-masing.

4.1.4 Struktur organisasi

Struktur organisasi perusahaan memiliki peran yang sangat penting dalam menjalankan kegiatan suatu perusahaan. Struktur organisasi perusahaan harus dapat meningkatkan mutu suatu perusahaan ke arah yang lebih baik dan dapat menciptakan koordinasi dan kerjasama diantara semua bagian yang terdapat dalam perusahaan dalam rangka meningkatkan kualitas kerja dan tanggung jawab. Adapun struktur organisasi *Software House* Lampung dapat dilihat pada Gambar 3.



Gambar 3. Struktur Organisasi *Software House* Lampung

4.2 Hasil dan pembahasan

Hasil dari tugas akhir yang berjudul “Aplikasi Pelaporan Keluhan Pelanggan Di *Software House* Lampung Berbasis *Website*” ini adalah :

4.2.1 Analisis

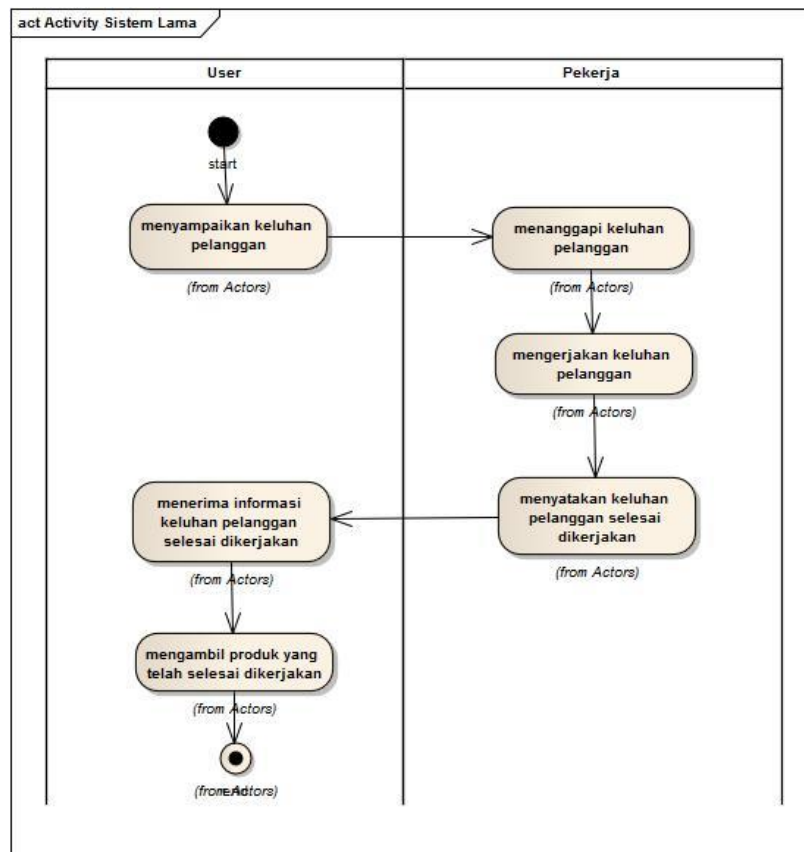
Tahap analisis bertujuan untuk mengetahui sistem yang sedang berjalan dan mengetahui kekurangan serta dapat memberikan solusi dari kekurangan pada sistem. Teknik yang digunakan dalam mengumpulkan data pada tahap ini adalah dengan cara wawancara dan observasi.

Narasumber yang memberikan informasi menjabat sebagai CTO yaitu Saudara Abraham Setyanugraha. Kemudian penulis melakukan observasi langsung pada saat Praktik Kerja Lapangan di *Software House* Lampung.

Hasil observasi terhadap sistem keluhan pelanggan yang sedang berjalan di *Software House* Lampung adalah:

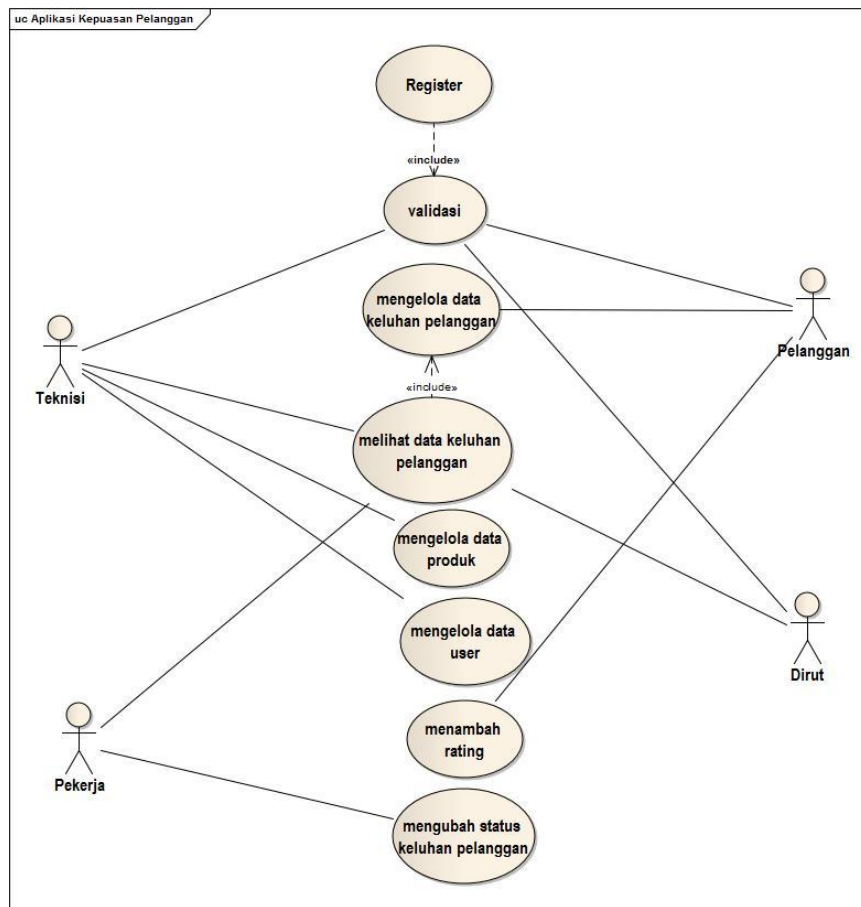
1. Pelanggan menyampaikan keluhan langsung kepada pekerja dengan datang ke *Software House* Lampung.
2. Pekerja menanggapi keluhan pelanggan tersebut.
3. Pekerja menyampaikan kepada pelanggan bahwa keluhannya telah dikerjakan melalui telepon.
4. Pelanggan mendapat informasi bahwa keluhannya telah dikerjakan dan mengambil produk yang telah selesai dikerjakan oleh pekerja di *Software House* Lampung.

Dari wawancara dan observasi yang dilakukan didapatkan beberapa informasi yaitu sejarah umum perusahaan, visi, misi, struktur organisasi *Software House* Lampung, dan sistem yang sedang berjalan. Berikut ini adalah *activity diagram* dari sistem yang sedang berjalan di *Software House* Lampung disajikan pada Gambar 4.



Gambar 4. Activity diagram sistem lama di *Software House* Lampung

Kebutuhan fungsional yaitu menjelaskan tentang proses-proses apa saja yang diberikan oleh sistem. Sistem ini terdapat tiga level yang memiliki hak akses masing-masing, yaitu level admin, pelanggan, dan pekerja. Berikut ini adalah *use case diagram* yang ada di *Software House* Lampung disajikan pada Gambar 5.



Gambar 5. Use case diagram

Kebutuhan fungsional pada level admin, pelanggan dan pekerja dijelaskan sebagai berikut:

1. Level admin (Teknisi)
 - a. Melihat data keluhan pelanggan.
 - b. Mengelola data produk.
 - c. Melihat dan mengubah data *user* atau pelanggan.
2. Level Dirut
 - a. Melihat data keluhan pelanggan.
3. Level *user* atau pelanggan
 - a. Mengelola data keluhan pelanggan.
 - b. Menambah rating.

4. Level Pekerja

- a. Melihat data keluhan pelanggan.
- b. Mengubah status keluhan pelanggan.

4.2.2 Desain

Berdasarkan analisis sistem pada tahap pertama kemudian dilakukan desain yang digunakan sebagai pengganti sistem yang sedang berjalan. Pada tahap ini perancangan sistem dilakukan dengan merancang *class diagram*, *sequence diagram*, *database*, dan *interface* aplikasi.

4.2.2.1 Desain sistem

Pada tahapan desain sistem membutuhkan *Unified Modelling Language* (UML) untuk mendeskripsikan proses-proses yang terjadi pada sistem yang akan diusulkan. UML merupakan suatu alat untuk menggambarkan kemudian membuat hasil analisa menjadi bentuk visual yang berisi sintak. Berikut ini tahapan diagram dalam UML yang digunakan dalam perancangan sistem yang diusulkan yaitu aplikasi pelaporan keluhan pelanggan di *Software House* Lampung terdiri dari *use case diagram* dan *sequence diagram*.

1. *Use case diagram*

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat atau sistem yang akan diusulkan. Pada aplikasi pelaporan keluhan pelanggan ini terdapat empat *use case diagram* yaitu *use case diagram* untuk admin, dirut, pelanggan dan pekerja yang telah disajikan pada Gambar 6.

a. *Use case diagram* level admin

Use case diagram level admin menggambarkan kelakuan atau tingkah laku yang admin lakukan di dalam sistem aplikasi pelaporan keluhan pelanggan.

Tingkah laku yang dilakukan oleh admin adalah login akun admin, melihat data keluhan pelanggan, mengelola data produk, dan mengelola data pelanggan.

b. *Use case diagram* level pelanggan

Use case diagram level pelanggan menggambarkan kelakuan atau tingkah laku yang pelanggan lakukan di dalam sistem aplikasi pelaporan keluhan pelanggan. Tingkah laku yang dilakukan oleh pelanggan adalah register akun pelanggan, mengelola data keluhan pelanggan, dan menambah rating kepuasan pelanggan.

c. *Use case diagram* level pekerja

Use case diagram level pekerja menggambarkan kelakuan atau tingkah laku yang pekerja lakukan di dalam sistem aplikasi pelaporan keluhan pelanggan. Tingkah laku yang dilakukan oleh pekerja adalah melihat data keluhan pelanggan dan mengubah status keluhan pelanggan.

2. Deskripsi *use case register*

Deskripsi *use case register* dari pelanggan pada aplikasi pelaporan keluhan pelanggan disajikan pada tabel 5.

Tabel 5. Deskripsi *use case register*

<i>Use case name</i>	<i>Register</i>	
<i>Actor</i>	Pelanggan	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk melakukan <i>register</i> dan dapat melakukan <i>login</i> ke dalam sistem.	
<i>Precondition</i>	<i>Actor</i> belum berhasil melakukan <i>register</i>	
<i>Pastcondition</i>	<i>Actor</i> berhasil melakukan <i>register</i>	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka Aplikasi 3. Mengisi <i>form register</i>	2. Menampilkan aplikasi 5. Jika <i>fullname</i> , <i>email</i> dan <i>password</i> benar, sistem akan menampilkan halaman <i>home</i>

3. Deskripsi *use case* validasi

Deskripsi pendefinisian *use case* dari tiga aktor pada aplikasi pelaporan keluhan pelanggan disajikan pada tabel 6.

Tabel 6. Deskripsi *use case* validasi

<i>Use case name</i>	Validasi	
<i>Actor</i>	Admin, Pelanggan, dan Pekerja	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk melakukan <i>login</i> ke dalam sistem.	
<i>Precondition</i>	<i>Actor</i> belum berhasil melakukan validasi	
<i>Pastcondition</i>	<i>Actor</i> berhasil melakukan validasi	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	4. Membuka Aplikasi 3. Mengisi <i>form login</i>	5. Menampilkan aplikasi 4. Cek validasi <i>email</i> dan <i>password</i> 5. Jika <i>email</i> dan <i>password</i> benar, sistem akan menampilkan halaman <i>home</i> 6. Jika <i>email</i> dan <i>password</i> salah, maka proses akan kembali ke no. 3

4. Deskripsi *use case* melihat data keluhan pelanggan

Deskripsi *use case* melihat data keluhan pelanggan merupakan penjabaran tentang alur dari proses melihat data keluhan pelanggan. Deskripsi *use case* melihat data keluhan pelanggan disajikan pada tabel 7.

Tabel 7. Deskripsi *use case* melihat data keluhan pelanggan

<i>Use case name</i>	Melihat data keluhan pelanggan	
<i>Actor</i>	Admin dan Pekerja	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk dapat melihat data keluhan pelanggan di dalam sistem.	
<i>Precondition</i>	Admin belum bisa melihat data keluhan pelanggan	
<i>Pastcondition</i>	Admin berhasil melihat data keluhan pelanggan	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>

Tabel 7. (Lanjutan)

	1. Membuka aplikasi	2. Menampilkan aplikasi
	3. Melakukan sesuai skenario pada tabel <i>login</i>	4. Menampilkan halaman utama (<i>home</i>)
	5. Melihat data keluhan pelanggan	6. Menampilkan halaman data keluhan pelanggan

5. Deskripsi *use case* mengelola data produk

Deskripsi *use case* mengelola data produk merupakan penjabaran tentang alur dari proses melihat, menambah, mengubah dan menghapus data produk. Deskripsi *use case* mengelola data produk disajikan pada tabel 8.

Tabel 8. Deskripsi *use case* mengelola data produk

<i>Use case name</i>	Mengelola data produk	
<i>Actor</i>	Admin	
<i>Description</i>	Actor menggunakan <i>use case</i> ini untuk dapat mengelola data produk di dalam sistem yaitu melihat, menambah, mengubah dan menghapus.	
<i>Precondition</i>	Admin belum bisa mengelola data produk	
<i>Pastcondition</i>	Admin berhasil mengelola data produk	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka aplikasi	2. Menampilkan aplikasi
	3. Melakukan sesuai skenario pada tabel <i>login</i>	4. Menampilkan halaman utama (<i>home</i>)
	5. Melihat data produk	6. Menampilkan halaman data produk
	7. Menambah data produk	8. Menampilkan <i>form</i> tambah produk
	9. Mengubah data produk	10. Menampilkan <i>form edit</i> produk
	11. Menghapus data produk	12. Menampilkan <i>form</i> hapus produk

6. Deskripsi *use case* mengelola data *user*

Deskripsi *use case* mengelola data *user* merupakan penjabaran tentang alur dari proses melihat, mengubah dan menghapus data *user*. Deskripsi *use case* mengelola data *user* disajikan pada tabel 9.

Tabel 9. Deskripsi *use case* mengelola data *user*

<i>Use case name</i>	Mengelola data <i>user</i>	
<i>Actor</i>	Admin	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk dapat mengelola data <i>user</i> di dalam sistem yaitu melihat, mengubah dan menghapus.	
<i>Precondition</i>	Admin belum bisa mengelola data <i>user</i>	
<i>Pastcondition</i>	Admin berhasil mengelola data <i>user</i>	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka aplikasi	2. Menampilkan aplikasi
	3. Melakukan sesuai skenario pada tabel <i>login</i>	4. Menampilkan halaman utama (<i>home</i>)
	5. Melihat data <i>user</i>	6. Menampilkan halaman data <i>user</i>
	7. Mengubah data <i>user</i>	8. Menampilkan <i>form edit user</i>
	9. Menghapus data <i>user</i>	10. Menampilkan <i>form hapus user</i>

7. Deskripsi *use case* mengelola data keluhan pelanggan

Deskripsi *use case* mengelola data keluhan pelanggan merupakan penjabaran tentang alur dari proses melihat, menambah, mengubah dan menghapus data keluhan pelanggan. Deskripsi *use case* mengelola data keluhan pelanggan akan disajikan pada tabel 10.

Tabel 10. Deskripsi *use case* mengelola data keluhan pelanggan

<i>Use case name</i>	Mengelola data keluhan pelanggan	
<i>Actor</i>	Pelanggan	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk dapat mengelola data keluhan pelanggan di dalam sistem yaitu melihat, menambah, mengubah dan menghapus.	
<i>Precondition</i>	Pelanggan belum bisa mengelola data keluhan pelanggan	
<i>Pastcondition</i>	Pelanggan berhasil mengelola data keluhan pelanggan	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka aplikasi	2. Menampilkan aplikasi
	4. Melakukan sesuai skenario pada tabel <i>login</i>	5. Menampilkan halaman utama (<i>home</i>)
	6. Melihat data keluhan pelanggan	7. Menampilkan halaman data keluhan pelanggan

Tabel 10. (Lanjutan)

	8. Menambah data keluhan pelanggan	9. Menampilkan <i>form</i> tambah keluhan pelanggan
	10. Mengubah data keluhan pelanggan	11. Menampilkan <i>form edit</i> keluhan pelanggan
	12. Menghapus data keluhan pelanggan	13. Menampilkan <i>form</i> hapus keluhan pelanggan

8. Deskripsi *use case* menambah *rating*

Deskripsi *use case* menambah *rating* keluhan pelanggan merupakan penjabaran tentang alur dari proses melihat, menambah, mengubas dan menghapus data keluhan pelanggan. Deskripsi *use case* mengelola data keluhan pelanggan akan disajikan pada tabel 11.

Tabel 11. Deskripsi *use case* menambah *rating*

<i>Use case name</i>	Menambah <i>rating</i>	
<i>Actor</i>	Pelanggan	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk dapat menambah <i>rating</i> kepuasan pelanggan di dalam sistem	
<i>Precondition</i>	Pelanggan belum bisa menambah <i>rating</i>	
<i>Pastcondition</i>	Pelanggan berhasil menambah <i>rating</i>	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka aplikasi	2. Menampilkan aplikasi
	3. Melakukan sesuai skenario pada tabel <i>login</i>	4. Menampilkan halaman utama (<i>home</i>)
	5. Melihat data keluhan pelanggan	6. Menampilkan halaman data keluhan pelanggan
	7. Menambah <i>rating</i> kepuasan pelanggan	8. Menampilkan halaman data keluhan pelanggan 9. Jika sudah menambah <i>rating</i> , maka akan menampilkan halaman data keluhan pelanggan yang sudah di <i>rating</i>

9. Deskripsi *use case* mengubah status keluhan pelanggan

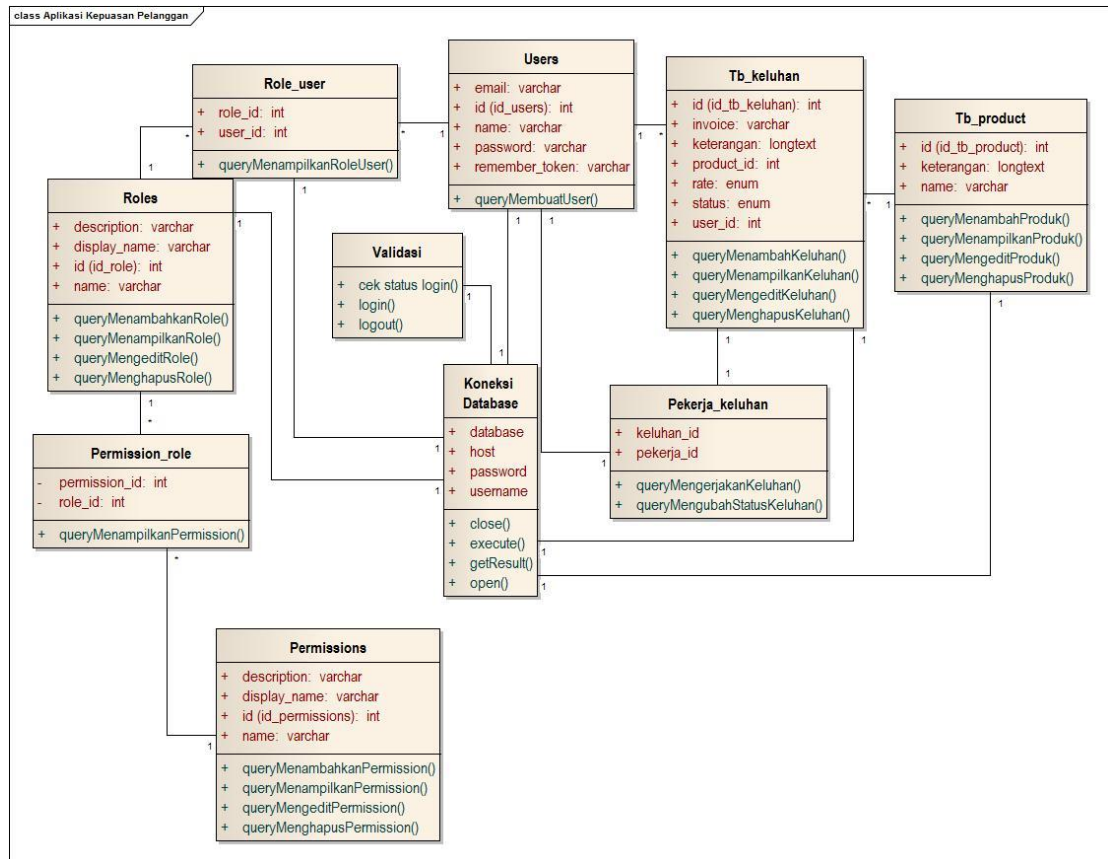
Deskripsi *use case* mengubah status keluhan pelanggan merupakan penjabaran tentang alur dari proses mengubah status keluhan pelanggan. Deskripsi *use case* mengubah status keluhan pelanggan disajikan pada tabel 12.

Tabel 12. Deskripsi *use case* mengubah status keluhan pelanggan

<i>Use case name</i>	Mengubah status keluhan pelanggan	
<i>Actor</i>	Pekerja	
<i>Description</i>	<i>Actor</i> menggunakan <i>use case</i> ini untuk dapat mengubah status pekerja di dalam sistem	
<i>Precondition</i>	Pekerja belum bisa menambah <i>rating</i>	
<i>Pastcondition</i>	Pekerja berhasil menambah <i>rating</i>	
<i>Basic flow</i>	<i>Actor</i>	<i>System</i>
	1. Membuka aplikasi	2. Menampilkan aplikasi
	3. Melakukan sesuai skenario pada tabel <i>login</i>	4. Menampilkan halaman utama (<i>home</i>)
	5. Melihat data keluhan pelanggan	6. Menampilkan halaman data keluhan pelanggan
	7. Mengubah status keluhan pelanggan	8. Menampilkan halaman data keluhan pelanggan 9. Jika sudah mengubah status, maka akan menampilkan halaman data keluhan pelanggan yang sudah diubah statusnya.

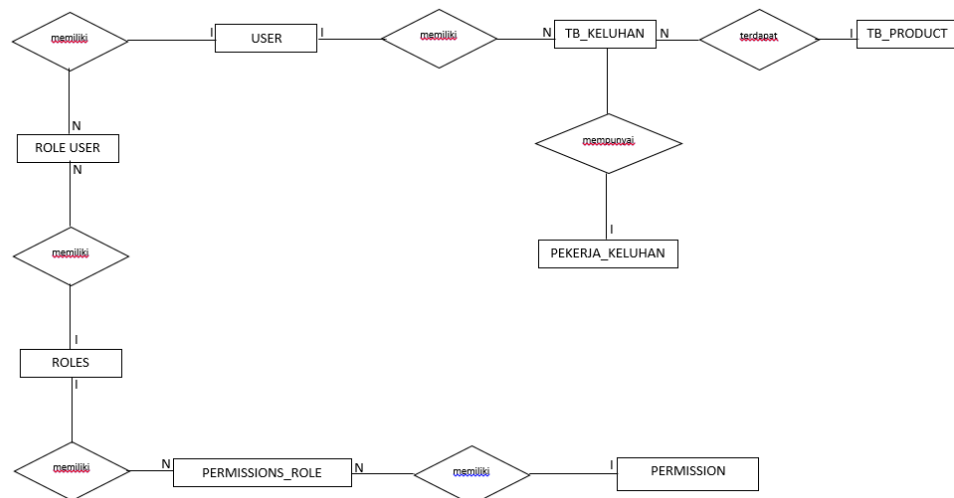
4.2.2.2 Desain *class diagram*

Desain *class diagram* yang dibuat melibatkan 9 *class* yaitu *class* Koneksi Database, *class* Users, *class* Tb_keluhan, *class* Tb_product, *class* Pekerja_keluhan, *class* role_user, *class* Roles, *class* Permission_role, *class* Permissions. Rancangan *class diagram* aplikasi pelaporan keluhan pelanggan disajikan pada Gambar 6.



Gambar 6. *Class diagram* aplikasi pelaporan keluhan pelanggan

Perancangan tabel *database* dilakukan dengan menyesuaikan *Entity Relationship Diagram* (ERD). ERD aplikasi pelaporan keluhan pelanggan disajikan pada Gambar 7.



Gambar 7. ERD aplikasi pelaporan keluhan pelanggan

Database dibuat menggunakan *software* MySQL. Terdapat 10 tabel pada *database* yaitu tabel *users*, tabel *tb_keluhan*, tabel *tb_product*, tabel *pekerja_keluhan*, tabel *roles*, tabel *role_user*, tabel *permissions*, tabel *permission-role*.

1. *Database* : laravel_admin.sql

Tabel : *users*

Primary Key : id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Id	<i>Integer</i>	10	-
Name	<i>Varchar</i>	30	-
Email	<i>Varchar</i>	25	-
Password	<i>Varchar</i>	10	-
Remember_token	<i>Varchar</i>	100	-

2. *Database* : laravel_admin.sql

Tabel : *tb_keluhan*

Primary Key : id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Id	<i>Integer</i>	10	-
Invoice	<i>Varchar</i>	10	-
User_id	<i>Integer</i>	10	-
Product_id	<i>Integer</i>	10	-
Keterangan	<i>Longtext</i>	-	-
Status	<i>Enum</i>	-	-
Rate	<i>Enum</i>	-	-

3. *Database* : laravel_admin.sql

Tabel : *tb_product*

Primary Key : id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Id	<i>Integer</i>	10	-
Name	<i>Varchar</i>	30	-
Keterangan	<i>Longtext</i>	-	-

4. *Database* : laravel_admin.sql

Tabel : pekerja_keluhan

Primary Key : -

Foreign Key : keluhan_id
pekerja_id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Keluhan_id	<i>Integer</i>	10	-
Pekerja_id	<i>integer</i>	10	-

5. *Database* : laravel_admin.sql

Tabel : roles

Primary Key : id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Id	<i>Integer</i>	10	-
Name	<i>Varchar</i>	30	-
Display_name	<i>Varchar</i>	30	-
Description	<i>Varchar</i>	30	-

6. *Database* : laravel_admin.sql

Tabel : role_user

Primary Key : -

Foreign Key : user_id
role_id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
User_id	<i>Integer</i>	10	-
Role_id	<i>Integer</i>	10	-

7. *Database* : laravel_admin.sql

Tabel : *permissions*

Primary Key : id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Id	<i>Integer</i>	10	-
Name	<i>Varchar</i>	30	-
Display_name	<i>Varchar</i>	30	-
Description	<i>Varchar</i>	30	-

8. *Database* : laravel_admin.sql

Tabel : *permission_role*

Primary Key : -

Foreign Key : *permission_id*
role_id

<i>Field Name</i>	<i>Data Type</i>	<i>Length</i>	<i>Ket</i>
Permission_id	<i>Integer</i>	10	-
Role_id	<i>Integer</i>	10	-

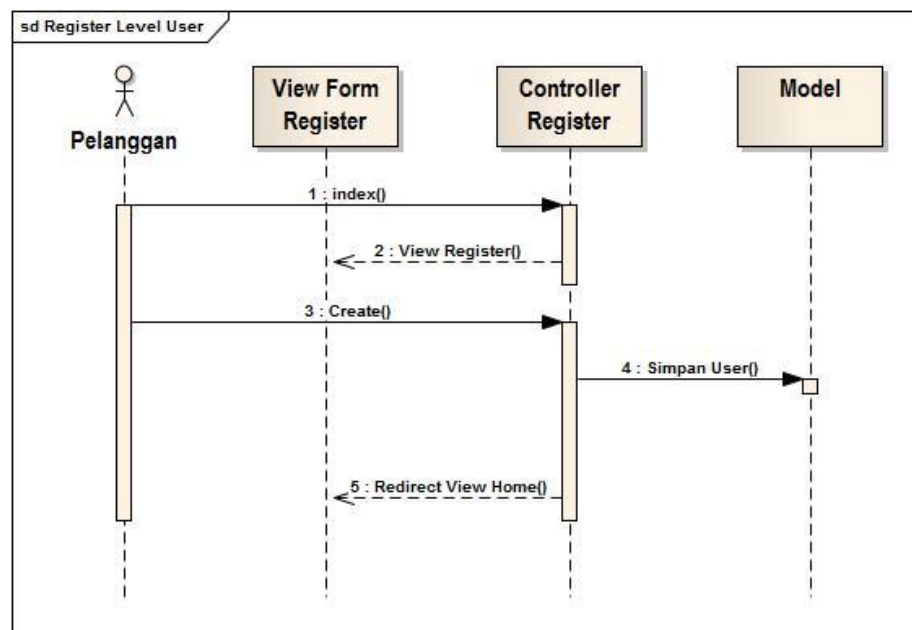
4.2.2.3 Desain sequence diagram

Sequence diagram adalah diagram yang digunakan untuk mendefinisikan *input* dan *output* serta urutan interaksi antara pengguna dan sistem untuk sebuah *use case*. *Sequence diagram* aplikasi pelaporan keluhan pelanggan menggunakan *Model-View-Controller* yaitu *view* untuk menampilkan informasi, *controller* sebagai penghubung maupun logika terhadap *view* dengan *model*, dan

model untuk mengambil dan memasukkan data ke *database*. *Sequence diagram* aplikasi pelaporan keluhan pelanggan diantaranya sebagai berikut:

1. *Sequence diagram* pelanggan register

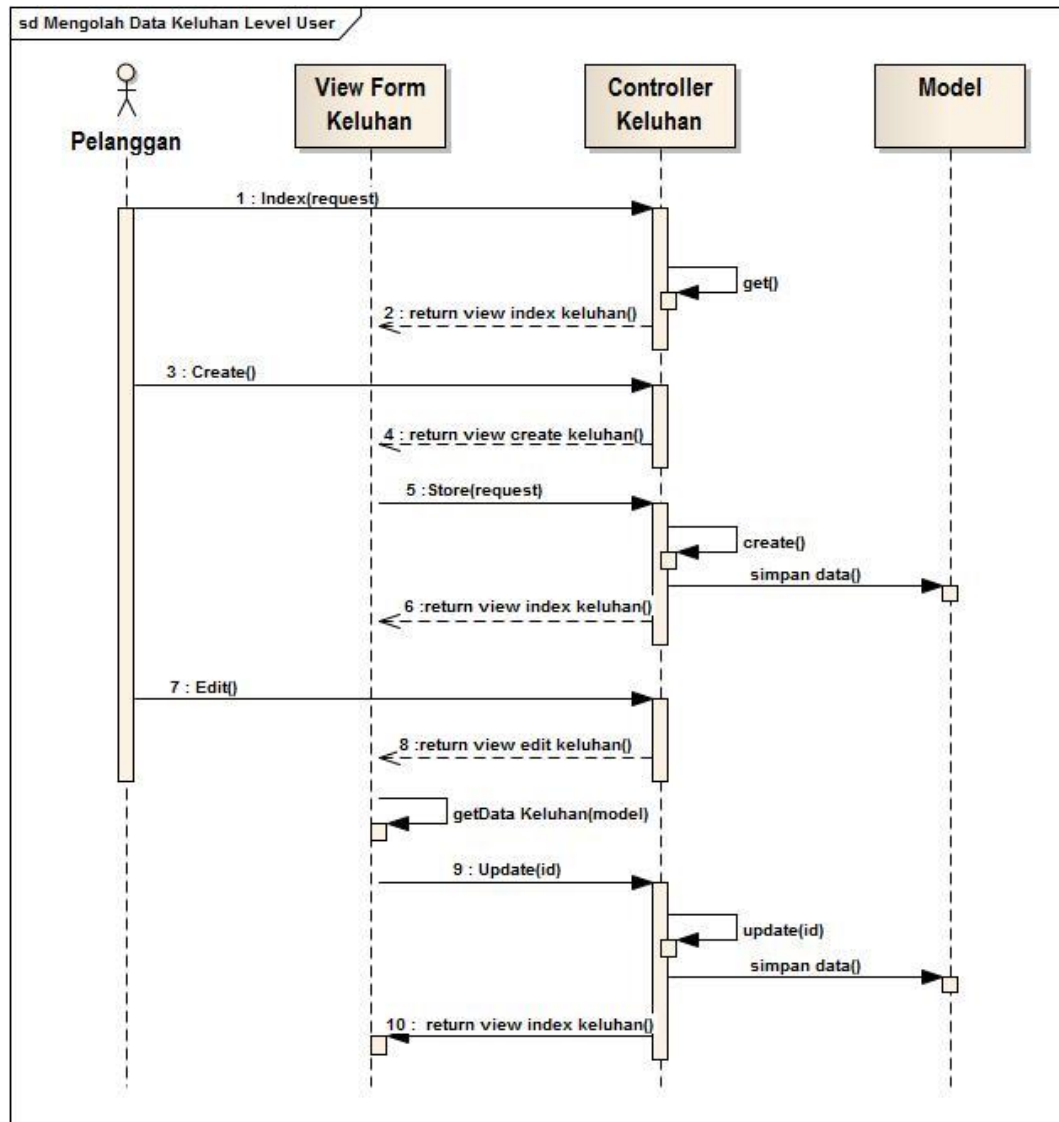
Sequence diagram untuk melakukan registrasi pendaftaran pelanggan menggambarkan alur proses di dalam pembuatan *member*. Rancangan *sequence diagram* melakukan registrasi pelanggan disajikan pada Gambar 8.



Gambar 8. *Sequence diagram* melakukan registrasi pelanggan

2. *Sequence diagram* mengelola data keluhan pelanggan

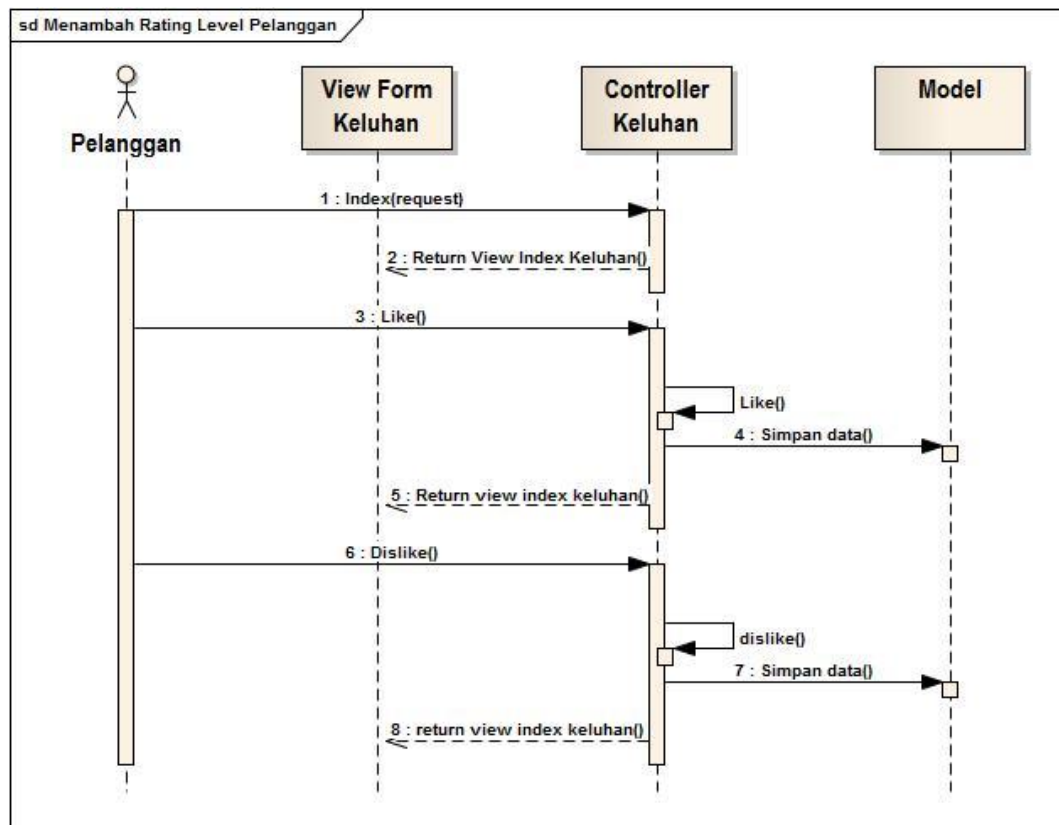
Sequence diagram mengelola data keluhan pelanggan menggambarkan alur proses dalam menambah, melihat, mengubah dan menghapus data keluhan pelanggan. Rancangan *sequence diagram* mengelola data keluhan pelanggan disajikan pada Gambar 9.



Gambar 9. *Sequence diagram* mengelola data keluhan pelanggan

3. *Sequence diagram* menambah rating

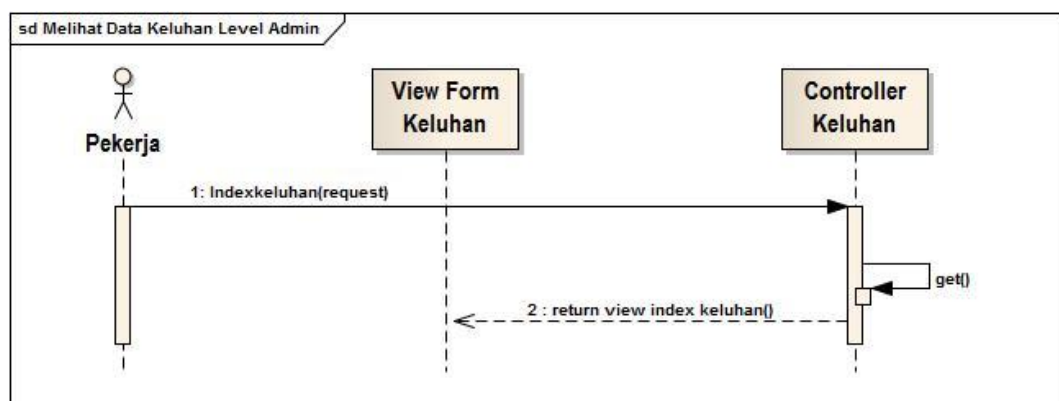
Sequence diagram menambah rating menggambarkan alur proses pelanggan menambah rating keluhan pelanggan. Rancangan *sequence diagram* menambah rating disajikan pada Gambar 10.



Gambar 10. *Sequence diagram* menambah rating

4. *Sequence diagram* melihat data keluhan pelanggan level admin

Sequence diagram melihat data keluhan pelanggan level admin menggambarkan alur proses dalam melihat data keluhan pelanggan pada aplikasi program. Rancangan *sequence diagram* melihat data keluhan pelanggan level admin disajikan pada Gambar 11.

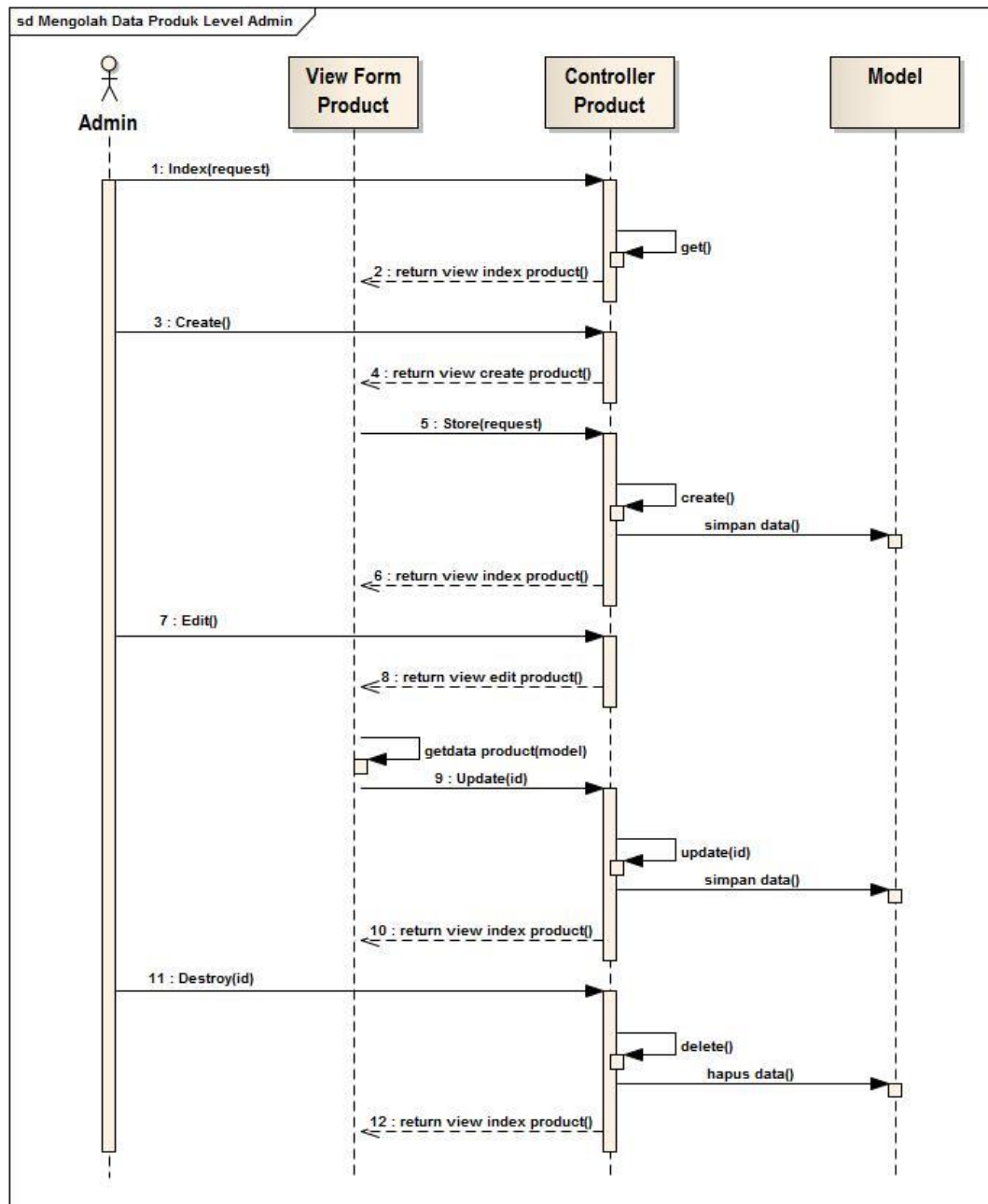


Gambar 11. *Sequence diagram* melihat data keluhan pelanggan level admin

5. *Sequence diagram* mengelola data produk level admin

Sequence diagram mengelola data produk level admin menggambarkan alur proses dalam menambah, melihat, mengubah dan menghapus data produk.

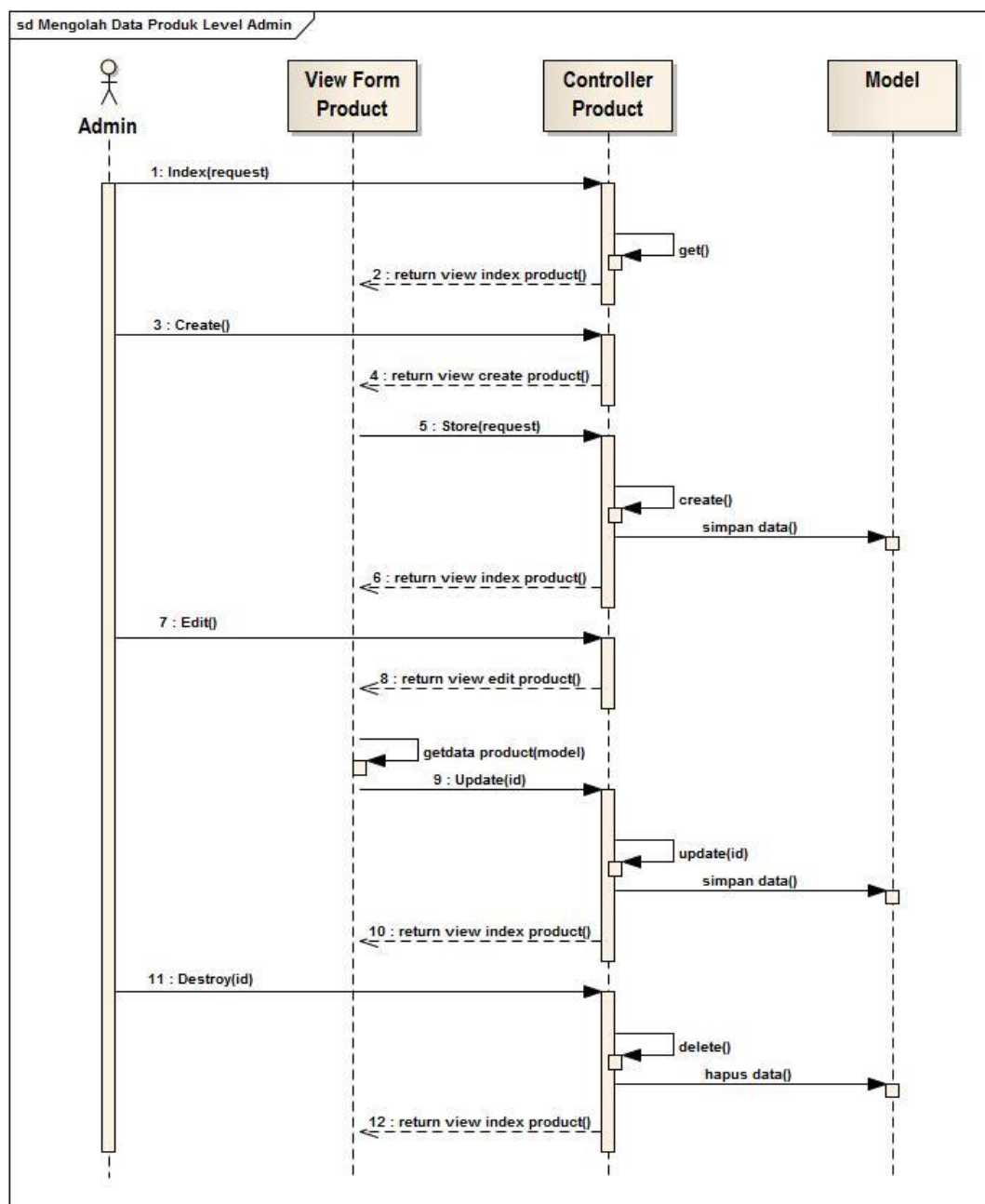
Rancangan *sequence diagram* mengelola data produk disajikan pada Gambar 12.



Gambar 12. *Sequence diagram* mengelola data produk level admin

6. *Sequence diagram* mengelola data *user* level admin

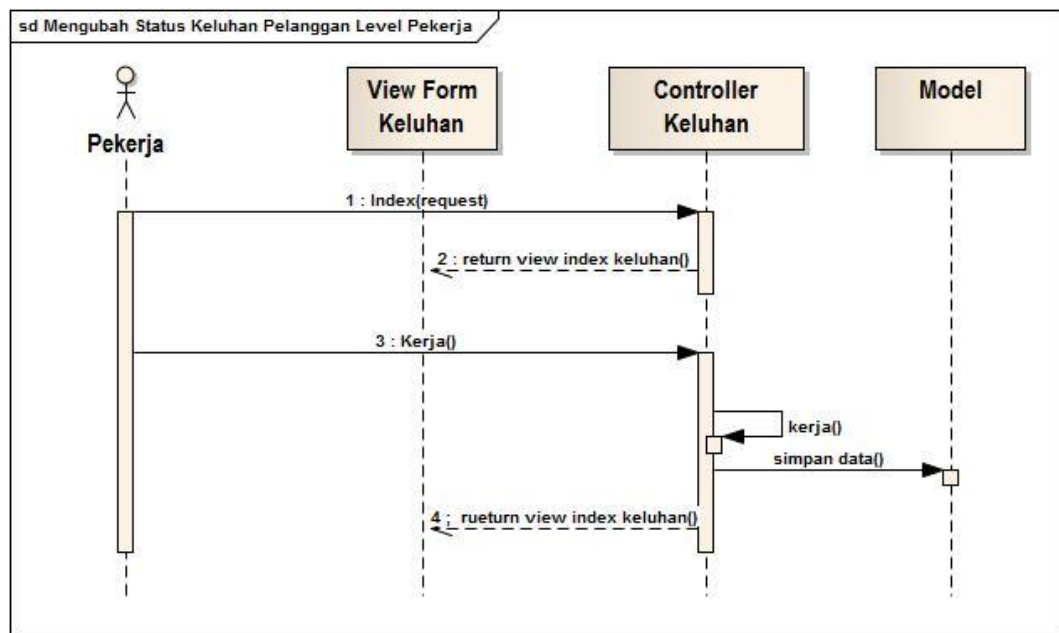
Sequence diagram mengelola data *user* level admin menggambarkan alur proses dalam melihat, mengubah dan menghapus data *user*. Rancangan *sequence diagram* mengelola data *user* disajikan pada Gambar 13.



Gambar 13. *Sequence diagram* mengelola data *user* level admin

7. *Sequence diagram* mengubah status keluhan pelanggan

Sequence diagram mengubah status keluhan pelanggan menggambarkan alur proses di dalam mengubah status pada aplikasi program. Rancangan *sequence diagram* mengubah status keluhan pelanggan disajikan pada Gambar 14.



Gambar 14. *Sequence diagram* mengubah status keluhan pelanggan level admin

4.2.2.4 Desain *interface*

Desain *interface* adalah rancangan *interface* yang akan diterapkan atau digunakan pada aplikasi yang akan dibuat. Desain *interface* dilakukan dengan harapan dapat memenuhi kebutuhan untuk aplikasi yang akan dibuat.

1. Rancangan tampilan *form login*

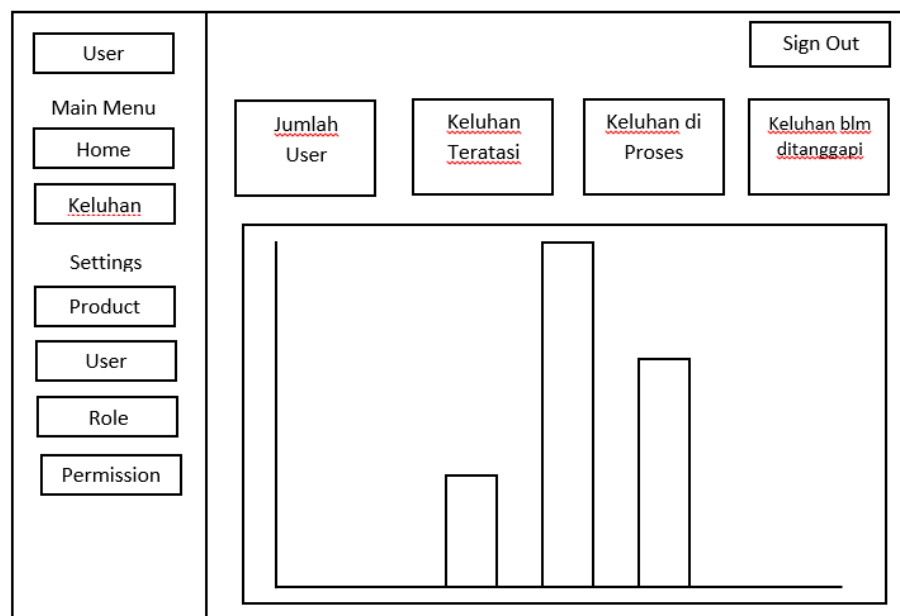
Form login terdiri dari kolom *username* dan *password* untuk akses masuk kedalam aplikasi, apabila *username* dan *password* benar dan sesuai dengan *database* maka *user* dapat masuk ke dalam aplikasi sesuai hak aksesnya. Rancangan tampilan *form login* disajikan pada Gambar 15.

The login form consists of a rectangular container with a border. Inside, there are five elements: an 'Email' input field, a 'Password' input field, a 'Remember me' checkbox, a 'Sign In' button, and a 'Register a new membership' button.

Gambar 15. Rancangan tampilan *form login*

2. Rancangan tampilan halaman utama admin

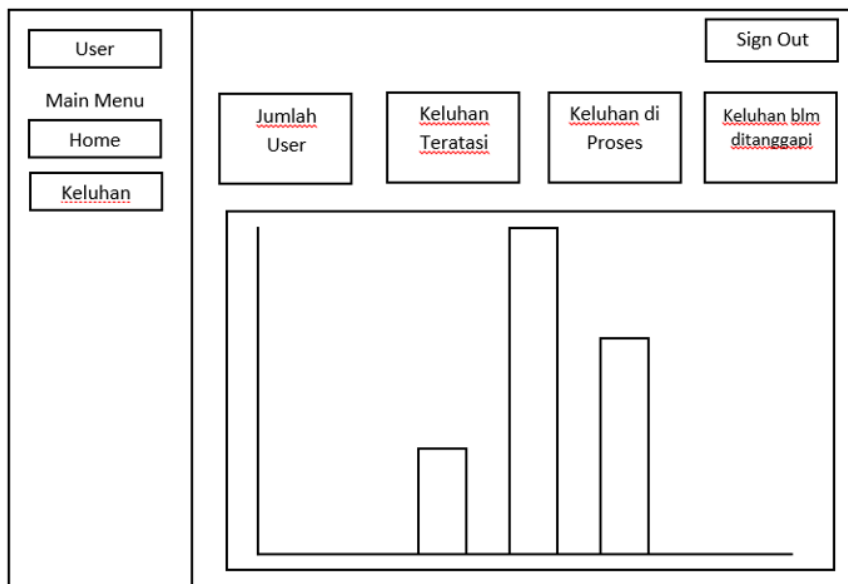
Halaman utama untuk *admin* yang terdiri dari beberapa menu yakni *user*, *home*, keluhan, *product*, *role*, *permission*, grafik, dan *logout*. Rancangan tampilan menu utama *admin* disajikan pada Gambar 16.



Gambar 16. Rancangan tampilan halaman utama admin

3. Rancangan tampilan halaman utama pelanggan

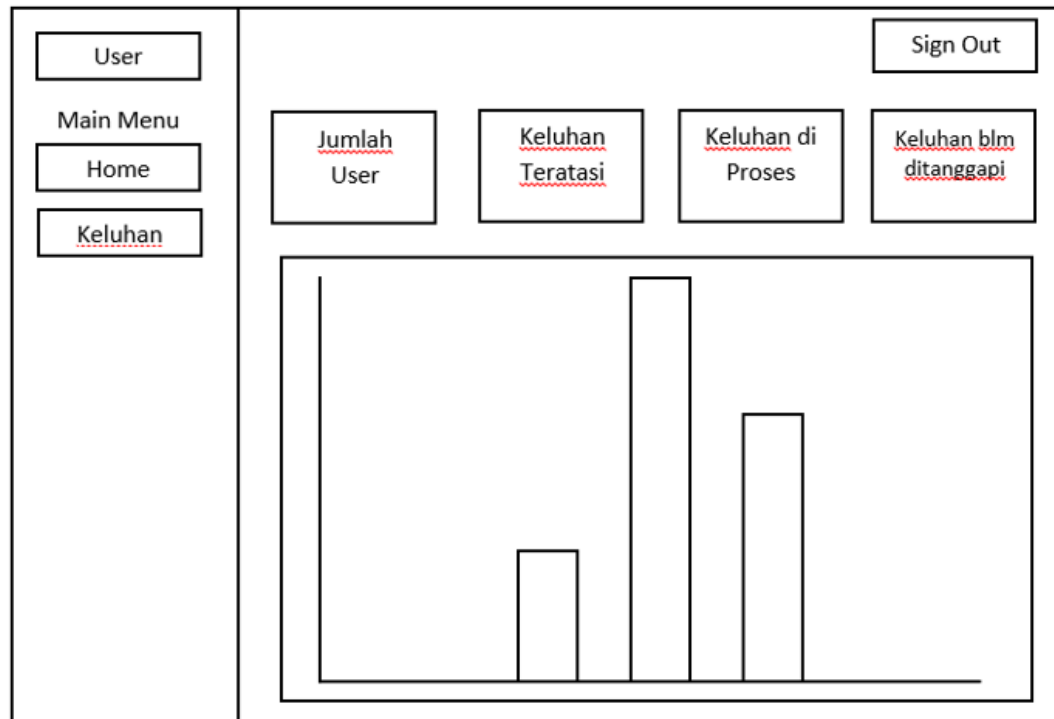
Halaman utama untuk pelanggan terdiri dari beberapa menu yakni *home*, keluhan, grafik, dan *logout*. Rancangan tampilan menu utama pelanggan disajikan pada Gambar 17.



Gambar 17. Rancangan tampilan halaman utama pelanggan

4. Rancangan tampilan halaman utama pekerja

Halaman utama untuk pekerja terdiri dari beberapa menu yakni *home*, keluhan, grafik, dan *logout*. Rancangan tampilan menu utama pekerja disajikan pada Gambar 18.



Gambar 18. Rancangan tampilan halaman utama pekerja

5. Rancangan *form register*

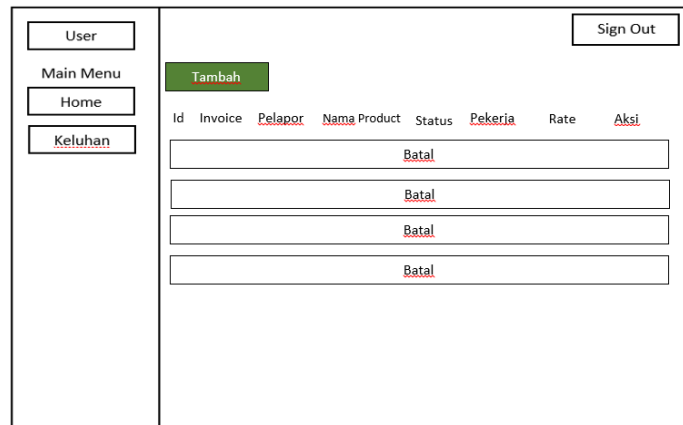
Form register adalah tampilan halaman untuk mendaftar atau membuat akun di halaman registrasi pada *website* aplikasi pelaporan keluhan pelanggan di *Software House* Lampung. Rancangan halaman *register* disajikan pada Gambar 19.

The figure shows a registration form design. It consists of four text input fields stacked vertically, labeled 'Full Name', 'Email', 'Password', and 'Password Confirmation'. A 'Register' button is positioned at the bottom right of the form area.

Gambar 19. Rancangan tampilan *form register*

6. Rancangan tampilan menu keluhan pelanggan

Tampilan menu keluhan pelanggan ini terdapat beberapa aksi seperti menambah, mengubah, dan menghapus. Rancangan tampilan menu keluhan pelanggan disajikan pada Gambar 20.

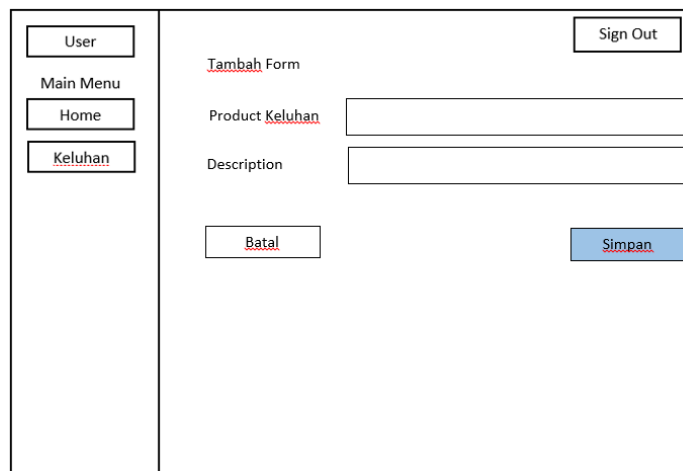


The image shows a web application interface for customer complaints. On the left is a sidebar menu with 'User', 'Main Menu', 'Home', and 'Keluhan' (highlighted). The top right has a 'Sign Out' button. The main content area has a green 'Tambah' button and a table with columns: Id, Invoice, Pelapor, Nama Product, Status, Pekerja, Rate, and Aksi. The table contains four rows, each with a 'Batal' link in the 'Aksi' column.

Gambar 20. Rancangan tampilan menu keluhan pelanggan

7. Rancangan tampilan *form* tambah keluhan pelanggan

Form tambah adalah tampilan halaman untuk menambah keluhan pelanggan di halaman *form* tambah pada *website* aplikasi pelaporan keluhan pelanggan di *Software House* Lampung. Rancangan tampilan *form* tambah keluhan pelanggan disajikan pada Gambar 21.



The image shows a web application interface for adding a new customer complaint. The sidebar menu is the same as in Gambar 20, with 'Keluhan' highlighted. The main content area has a 'Sign Out' button and a section titled 'Tambah Form'. It contains two input fields: 'Product Keluhan' and 'Description'. At the bottom, there are two buttons: 'Batal' and 'Simpan'.

Gambar 21. Rancangan tampilan *form* tambah keluhan pelanggan

8. Rancangan tampilan *form edit* keluhan pelanggan

Form edit adalah tampilan halaman untuk mengubah keluhan pelanggan di halaman *form edit* pada *website* aplikasi pelaporan keluhan pelanggan di *Software House* Lampung. Rancangan tampilan *form edit* keluhan pelanggan disajikan pada Gambar 22.

<div>User</div> <div>Main Menu</div> <div>Home</div> <div>Keluhan</div>	<div>Sign Out</div> <div>Edit Form</div> <div>Product <u>Keluhan</u></div> <div>Description</div> <div>Batal</div> <div>Simpan</div>
---	--

Gambar 22. Rancangan tampilan *form edit* keluhan pelanggan

4.2.3 Pengodean

4.2.3.1 Pembuatan Kode Program

Pengodean merupakan tahap penerapan dari hasil analisis dan desain untuk diterjemahkan ke dalam bahasa komputer. Pada pengodean yang menggunakan metode sistem *Model-View-Controller* (MVC). Pengodean pada “Aplikasi Pelaporan Keluhan Pelanggan di *Software House* Lampung berbasis *website*” dilakukan pengodean menggunakan aplikasi *notepad++*. Kode program yang ditampilkan hanya kode program pada bagian *controller* saja.

1. Penulisan kode program pada *HomeController.php*

```

<?php
namespace App\Http\Controllers;
use App\Keluhan;
use App\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
class HomeController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth');
    }
    public function index()
    {
        $cuser = User::all();
        if (Auth::user()->hasRole('admin')) {
            $sks = Keluhan::where('status', 3)->get();
            $skp = Keluhan::where('status', 2)->get();
            $skb = Keluhan::where('status', 1)->get();
            for ($i = 1; $i <= 12; $i++) {
                $schks[$i] = Keluhan::where('status', 3)
                    ->whereMonth('created_at', '=', $i)
                    ->whereYear('created_at', '=', date('Y'))->count();
                $schkp[$i] = Keluhan::where('status', 2)
                    ->whereMonth('created_at', '=', $i)
                    ->whereYear('created_at', '=', date('Y'))->count();
                $schkb[$i] = Keluhan::where('status', 1)
                    ->whereMonth('created_at', '=', $i)
                    ->whereYear('created_at', '=', date('Y'))->count();
            }
        } else {
            if (Auth::user()->hasRole('pelanggan')) {
                $sks = Keluhan::where('status', 3)->where('user_id', Auth::user()->id)->get();
                $skp = Keluhan::where('status', 2)->where('user_id', Auth::user()->id)->get();
                $skb = Keluhan::where('status', 1)->where('user_id', Auth::user()->id)->get();
                for ($i = 1; $i <= 12; $i++) {
                    $schks[$i] = Keluhan::where('status', 3)
                        ->where('user_id', Auth::user()->id)
                        ->whereMonth('created_at', '=', $i)
                        ->whereYear('created_at', '=', date('Y'))->count();
                    $schkp[$i] = Keluhan::where('user_id', Auth::user()->id)
                        ->where('status', 2)
                        ->whereMonth('created_at', '=', $i)

```

```

        ->whereYear('created_at', '=', date('Y'))->count();
        $chkb[$i] = Keluhan::where('user_id', Auth::user()->id)
        ->where('status', 1)
        ->whereMonth('created_at', '=', $i)
        ->whereYear('created_at', '=', date('Y'))->count();
    }
} else {
    $cks = Keluhan::where('status', 3)->get();
    $ckp = Keluhan::where('status', 2)->get();
    $ckb = Keluhan::where('status', 1)->get();
    for ($i = 1; $i <= 12; $i++) {
        $chks[$i] = Keluhan::where('status', 3)
        ->whereMonth('created_at', '=', $i)
        ->whereYear('created_at', '=', date('Y'))->count();
        $chkp[$i] = Keluhan::where('status', 2)
        ->whereMonth('created_at', '=', $i)
        ->whereYear('created_at', '=', date('Y'))->count();
        $chkb[$i] = Keluhan::where('status', 1)
        ->whereMonth('created_at', '=', $i)
        ->whereYear('created_at', '=', date('Y'))->count();
    }
}
return view('home', compact('cuser', 'cks', 'ckp', 'ckb', 'chks', 'chkp',
'chkb'));
}
}

```

2. Penulisan kode program pada *KeluhanController.php*

```

<?php
namespace App\Http\Controllers;
use App\Keluhan;
use App\PekerjaKeluhan;
use App\Product;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use PDF;
class KeluhanController extends Controller
{
    public function index()
    {
        if(Auth::user()->hasRole('pelanggan')) {
            $keluhan = Keluhan::where('user_id', Auth::user()->id)->get();
        } else {
            $keluhan = Keluhan::all();
        }
        return view('keluhan.index', compact ('keluhan'));
    }
}

```

```

public function create()
{
    $product = Product::all();
    return view('keluhan.tambah', compact('product'));
}
public function store(Request $request)
{
    $this->validate($request, [
        'productkeluhan' => 'required',
        'keterangan' => 'required',
    ]);
    $tambah = new Keluhan();
    $inis = Product::find($request['productkeluhan']);
    $arr = explode(' ', $inis->name);
    $str = substr($arr[0], -3);
    $ini = strtoupper($str);
    $tambah->invoice = $ini.$request['productkeluhan'].Auth::user()->id;
    $tambah->user_id = Auth::user()->id;
    $tambah->product_id = $request['productkeluhan'];
    $tambah->keterangan = $request['keterangan'];
    $tambah->save();
    $request->session()->flash('message', 'Data berhasil ditambahkan');
    return redirect()->to('/keluhan');}
public function show($id)
{
}
public function edit($id)
{
    $show = Keluhan::find($id);
    $product = Product::all();
    return view('keluhan.detail', compact('show', 'product'));
}
public function update(Request $request, $id)
{
    $this->validate($request, [
        'keterangan' => 'required',
    ]);
    $update = Keluhan::find($id);
    $update->keterangan = $request['keterangan'];
    $update->update();
    return redirect()->to('/keluhan');
}
public function destroy($id)
{
    $destroy = Keluhan::where('id', $id)->first();
    $destroy->delete();
    \Session::flash('message', 'Data berhasil dihapus');
    return redirect()->to('/keluhan');
}

```

```

public function pdf($id)
{
    $show = Keluhan::find($id);
    $product = Product::all();
    $pdf = PDF::loadView('keluhan.pdf', compact('show','product'));
    return $pdf->stream();
}
public function kerja($id)
{
    $kerja = Keluhan::where('id', $id)->first();
    if($kerja->status == 1) {
        $tambah = new PekerjaKeluhan();
        $tambah->keluhan_id = $id;
        $tambah->pekerja_id = Auth::user()->id;
        $tambah->save();
        $kerja->status = 2;
    } else {
        $kerja->status = 3;
    }
    $kerja->update();
    \Session::flash('message', 'Data berhasil diubah');
    return redirect()->to('/keluhan');
}
public function dislike($id)
{
    $kerja = Keluhan::where('id', $id)->first();
    $kerja->rate = 2;
    $kerja->update();
    \Session::flash('message', 'Data berhasil diubah');
    return redirect()->to('/keluhan');
}
public function like($id)
{
    $kerja = Keluhan::where('id', $id)->first();
    $kerja->rate = 3;
    $kerja->update();
    \Session::flash('message', 'Data berhasil diubah');
    return redirect()->to('/keluhan');
}
}
}

```

3. Penulisan kode program pada *PermissionController.php*

```

<?php
namespace App\Http\Controllers;
use App\Permission;
use App\PermissionRole;

```

```

use App\Role;
use Illuminate\Http\Request;
class PermissionController extends Controller
{
    public function index()
    {
        $roles = Role::all();
        $perms = Permission::all();
        return view('permission.index', compact('perms', 'roles'));
    }
    public function create()
    {
        return view('permission.tambah');
    }
    public function store(Request $request)
    {
        $this->validate($request, [
            'name' => 'required',
            'display_name' => 'required',
            'description' => 'required',
        ]);
        $tambah = new Permission();
        $tambah->name = $request['name'];
        $tambah->display_name = $request['display_name'];
        $tambah->description = $request['description'];
        $tambah->save();
        return redirect()->to('/permission');
    }
    public function show($id)
    {}
    public function edit($id)
    {
        $show = Permission::find($id);
        return view('permission.detail', compact('show'));
    }
    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'name' => 'required',
            'display_name' => 'required',
            'description' => 'required',
        ]);
        $update = Permission::find($id);
        $update->name = $request['name'];
        $update->display_name = $request['display_name'];
        $update->description = $request['description'];
        $update->update();
        return redirect()->to('/permission');
    }
}

```



```

    }
    public function destroy($id)
    {
        $hapus = Permission::find($id);
        $hapus->delete();
        return redirect()->to('/permission');
    }
    public function makePermiRole($perm, $role)
    {
        $pr = new PermissionRole();
        $pr->permission_id = $perm;
        $pr->role_id = $role;
        $pr->save();
    }
    public function delePermiRole($perm, $role)
    {
        $pr = PermissionRole::where([
            ['permission_id', '=', $perm],
            ['role_id', '=', $role]]);
        $pr->delete();
    }
}

```

4. Penulisan kode program pada *ProductController.php*

```

<?php
namespace App\Http\Controllers;
use App\Product;
use Illuminate\Http\Request;
class ProductController extends Controller
{
    public function index()
    {
        $product = Product::all();
        return view('product.index', compact('product'));
    }
    public function create()
    {
        return view('product.tambah');
    }
    public function store(Request $request)
    {
        $this->validate($request, [
            'name' => 'required',
            'description' => 'required',
        ]);
        $tambah = new Product();
    }
}

```

```

        $tambah->name = $request['name'];
        $tambah->keterangan = $request['description'];
        $tambah->save();
        return redirect()->to('/product');
    }
    public function show($id)
    {}
    public function edit($id)
    {
        $show = Product::find($id);
        return view('product.detail',compact('show'));
    }
    public function update(Request $request, $id)
    {
        $this->validate($request, [
            'name' => 'required',
            'description' => 'required',
        ]);
        $update = Product::find($id);
        $update->name = $request['name'];
        $update->keterangan = $request['description'];
        $update->update();
        return redirect()->to('/product');
    }
    public function destroy($id)
    {
        $hapus = Product::find($id);
        $hapus->delete();
        return redirect()->to('/product');
    }
}

```

5. Penulisan kode program pada *RoleController.php*

```

<?php
namespace App\Http\Controllers;
use App\Role;
use Illuminate\Http\Request;
class RoleController extends Controller
{
    public function index()
    {
        $role = Role::all();
        return view('role.index', compact('role'));
    }
    public function create()

```

```

{
    return view('role.tambah');
}
public function store(Request $request)
{
    $this->validate($request, [
        'name' => 'required',
        'display_name' => 'required',
        'description' => 'required',
    ]);
    $tambah = new Role();
    $tambah->name = $request['name'];
    $tambah->display_name = $request['display_name'];
    $tambah->description = $request['description'];
    $tambah->save();
    return redirect()->to('/role');
}
public function show($id)
{
}
public function edit($id)
{
    $show = Role::find($id);
    return view('role.detail',compact('show'));
}
public function update(Request $request, $id)
{
    $this->validate($request, [
        'name' => 'required',
        'display_name' => 'required',
        'description' => 'required',
    ]);
    $update = Role::find($id);
    $update->name = $request['name'];
    $update->display_name = $request['display_name'];
    $update->description = $request['description'];
    $update->update();
    return redirect()->to('/role');
}
public function destroy($id)
{
    $role = Role::findOrFail($id); // Pull back a given role
    // Regular Delete
    $role->delete(); // This will work no matter what
    // Force Delete
    //$role->users()->sync([]); // Delete relationship data
    //$role->perms()->sync([]); // Delete relationship data
    //$role->forceDelete(); // Now force delete will work regardless of
    // whether the pivot table has cascading delete
}

```

```

        return redirect()->to('/role');
    }}

```

6. Penulisan kode program pada *UserController.php*

```

<?php
namespace App\Http\Controllers;
use App\Role;
use App\RoleUser;
use App\User;
use Illuminate\Http\Request;
class UserController extends Controller
{
    public function index()
    {
        $user = User::all();
        $roles = Role::all();
        return view('user.index', compact('user', 'roles'));
    }
    public function create()
    {
        return view('user.tambah');
    }
    public function store(Request $request)
    {
        $this->validate($request, [
            'nama' => 'required',
            'email' => 'required',
            'password' => 'required',
        ]);
        $tambah = new User();
        $tambah->name = $request['nama'];
        $tambah->email = $request['email'];
        $tambah->password = $request['password'];
        $tambah->save();
        return redirect()->to('/user');
    }
    public function show($id)
    {
    }
    public function edit($id)
    {
        $show = User::find($id);
        $roles = Role::all();
        return view('user.detail', compact('show', 'roles'));
    }
    public function update(Request $request, $id)

```

```

{
    $update = User::where('id', $id)->first();
    $update->name = $request['name'];
    $update->email = $request['email'];
    $update->update();
    return redirect()->to('/user');
}
public function destroy($id)
{
    $hapus = User::find($id);
    $hapus->delete();
    return redirect()->to('/user');
}
public function makeUserRole($user, $role)
{
    $ru = new RoleUser();
    $ru->user_id = $user;
    $ru->role_id = $role;
    $ru->save();
}
public function deleteUserRole($user, $role)
{
    $ru = RoleUser::where([
        ['user_id', '=', $user],
        ['role_id', '=', $role]]);
    $ru->delete();
}}


```

4.2.3.2 Tampilan Aplikasi

Pada tahapan ini akan ditampilkan beberapa tampilan program yang dibuat berdasarkan rancangan-rancangan yang telah diusulkan pada tahap sebelumnya :

1. Tampilan *form login*

Form login berfungsi untuk mendapatkan hak akses, untuk dapat mengakses aplikasi tersebut admin, pelanggan dan pekerja harus memasukkan *email* dan *password*. Tampilan *form login* disajikan pada Gambar 23.



**Software House
Lampung**

Sign in to start your session

Email

Password

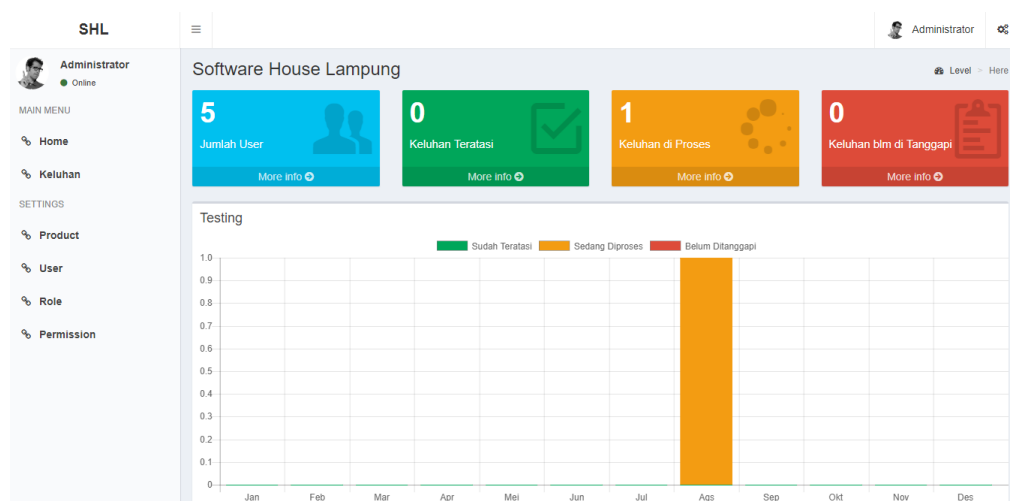
☐ Remember Me Sign In

[Register a new membership](#)

Gambar 23. Tampilan *form login*

2. Tampilan halaman utama admin

Halaman utama admin menampilkan grafik keluhan pelanggan, dan menampilkan menu yang dapat dilakukan oleh admin yaitu menu *home*, keluhan, *user*, *product*, *role*, *permission*. Tampilan halaman utama admin disajikan pada Gambar 24.



Gambar 24. Tampilan halaman utama admin

3. Tampilan halaman menu keluhan pelanggan level admin

Halaman menu keluhan pelanggan pada level admin berfungsi untuk menampilkan data keluhan pelanggan dengan dapat menghapus dan mencetak

laporan keluhan pelanggan. Tampilan halaman menu keluhan pelanggan disajikan pada Gambar 25.

Id	Invoice	Pelapor	Nama Product	Status	Pekerja	Rate	Aksi
2	ING85	Ochi Febrianti	Hosting	Sedang di proses	Abraham Setyanugraha	unrated	Hapus PDF
3	ITE14	Yoki Satria	Website Company Profile	Masalah selesai	Abraham Setyanugraha	Dislike Like	Hapus PDF
4	TAL24	Yoki Satria	Portal Berita Online	Belum di tanggapi		unrated	Hapus PDF

Gambar 25. Tampilan halaman menu keluhan pelanggan

4. Tampilan halaman menu *product* level admin

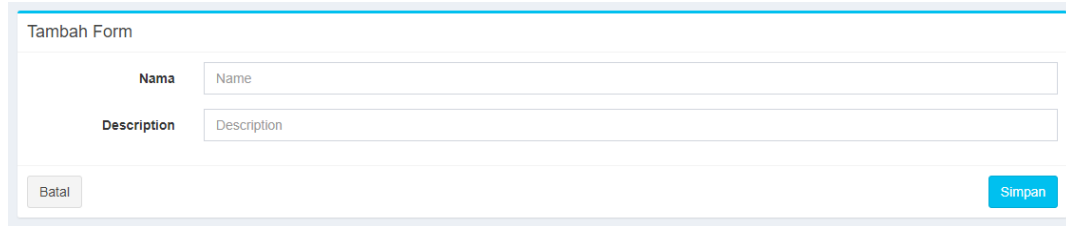
Halaman menu *product* pada level admin berfungsi untuk menampilkan data *product* dengan dapat menambah, mengedit dan menghapus data *product*. Tampilan halaman menu *product* disajikan pada Gambar 26.

Id	Nama Product	Deskripsi Product	Aksi
1	Website Company Profile	Website Company Profile	Hapus Edit
2	Portal Berita Online	Portal Berita Online	Hapus Edit
3	Toko Online	Toko Online	Hapus Edit
4	Software Bisnis	Software Bisnis	Hapus Edit
5	Software Akunting	Software Akunting	Hapus Edit
6	Jasa SEO	Jasa SEO	Hapus Edit
7	Internet Marketing	Internet Marketing	Hapus Edit
8	Hosting	Hosting	Hapus Edit
9	Domain	Domain	Hapus Edit
10	Custom	Custom	Hapus Edit

Gambar 26. Tampilan halaman menu *product*

5. Tampilan *form* tambah *product*

Tampilan *form* tambah *product* digunakan untuk menambah data *product*. Tampilan *form* tambah *product* disajikan pada Gambar 27.

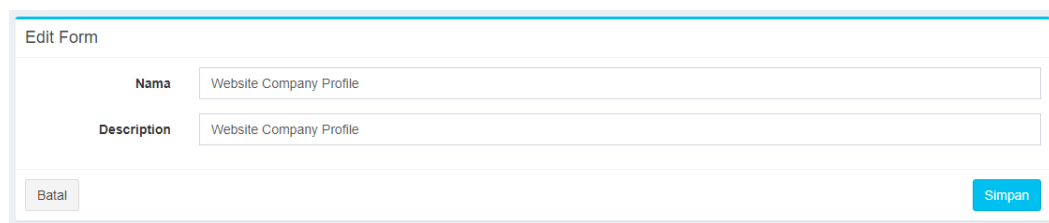


Gambar 27. Tampilan *form* tambah *product*

6. Tampilan *form edit product*

Tampilan *form edit product* digunakan untuk mengubah data *product*.

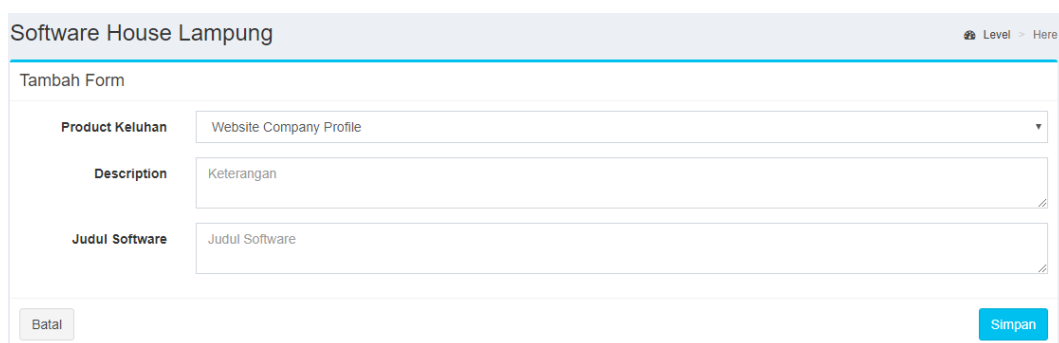
Tampilan *form edit product* disajikan pada Gambar 28.



Gambar 28. Tampilan *form edit product*

7. Tampilan *form* tambah keluhan pelanggan level pelanggan

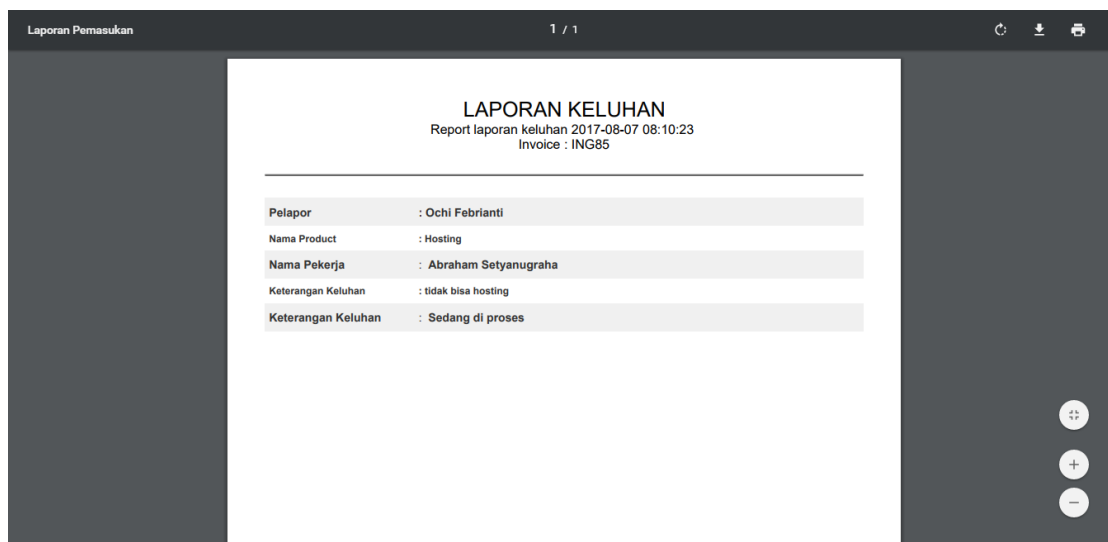
Tampilan *form* tambah keluhan pelanggan digunakan untuk menambah data keluhan pelanggan. Tampilan *form* tambah keluhan pelanggan disajikan pada Gambar 29.



Gambar 29. Tampilan *form* tambah keluhan pelanggan

8. Tampilan halaman *report* keluhan pelanggan

Halaman *report* keluhan pelanggan digunakan untuk laporan keluhan pelanggan pada saat melaporkan keluhannya. Tampilan halaman *report* keluhan pelanggan disajikan pada Gambar 30.



Gambar 30. Tampilan halaman *report* keluhan pelanggan

4.2.4 Pengujian Sistem

Tahap pengujian sistem secara lengkap dilakukan untuk menjamin bahwa syarat dan spesifikasi sistem telah terpenuhi berdasarkan persyaratan-persyaratan yang didapat pada tahap selanjutnya.

4.2.4.1 Metode Pengujian

Metode pengujian yang digunakan pada “Aplikasi Pelaporan Keluhan Pelanggan di *Software House* Lampung” adalah *black box testing*, yaitu pengujian dilakukan dengan menjalankan unit atau modul untuk mengamati apakah terjadi kesalahan atau sudah sesuai dengan kebutuhan sistem yang telah dibuat.

4.2.4.2 Hal-hal yang diuji

Hal-hal yang diuji dalam tugas akhir yang berjudul “Aplikasi Pelaporan Keluhan Pelanggan di *Software House* Lampung” ini adalah sebagai berikut :

1. Kesalahan-kesalahan pada tampilan
2. Kesalahan basis data

4.2.4.3 Hasil Pengujian

Setelah melakukan langkah-langkah pengujian di atas, aplikasi yang telah diuji berdasarkan lampiran 1 dapat disimpulkan bahwa aplikasi ini siap untuk digunakan dalam proses pelaporan keluhan pelanggan di *Software House* Lampung serta mempermudah atasan untuk memantau keluhan pelanggan yang ada dan pekerja. Aplikasi ini dapat mempermudah dan menghemat waktu pelanggan dalam melaporkan keluhannya.

4.2.5 Pendukung Sistem

Tahap pendukung sistem atau pemeliharaan ini belum dapat dilakukan karena aplikasi pelaporan keluhan pelanggan di *Software House* Lampung belum dijalankan.