**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*.

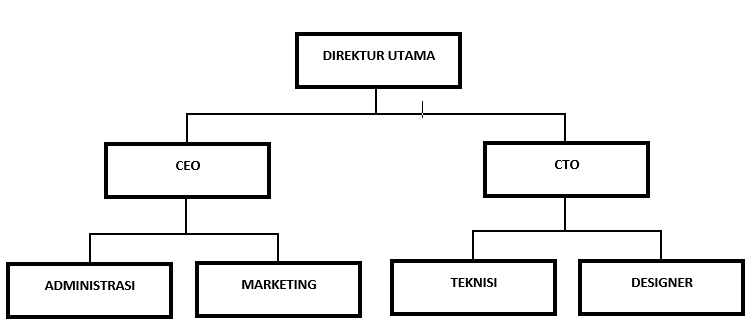
**4.1.3 Misi perusahaan**

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**

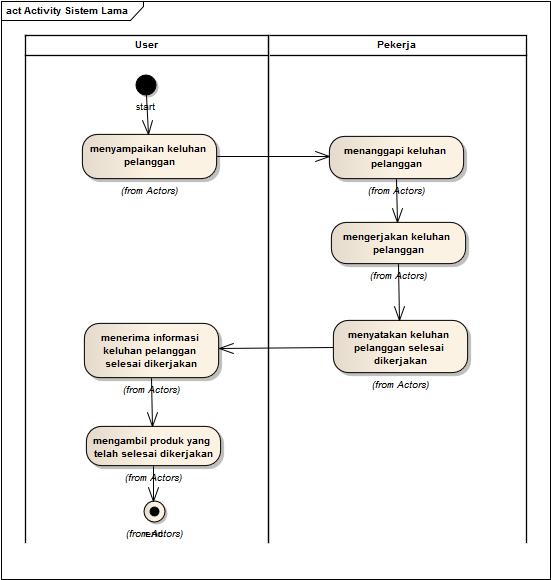
Tahap analisis bertujuan untuk mengetahui sistem yang sedang berjalan dan mengetahui kekurangan serta dapat memberikan solusi dari kekurangan pada sistem. Teknik yang digunakan dalam mengumpulkan data pada tahap ini adalah dengan cara wawancara dan observasi.

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

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

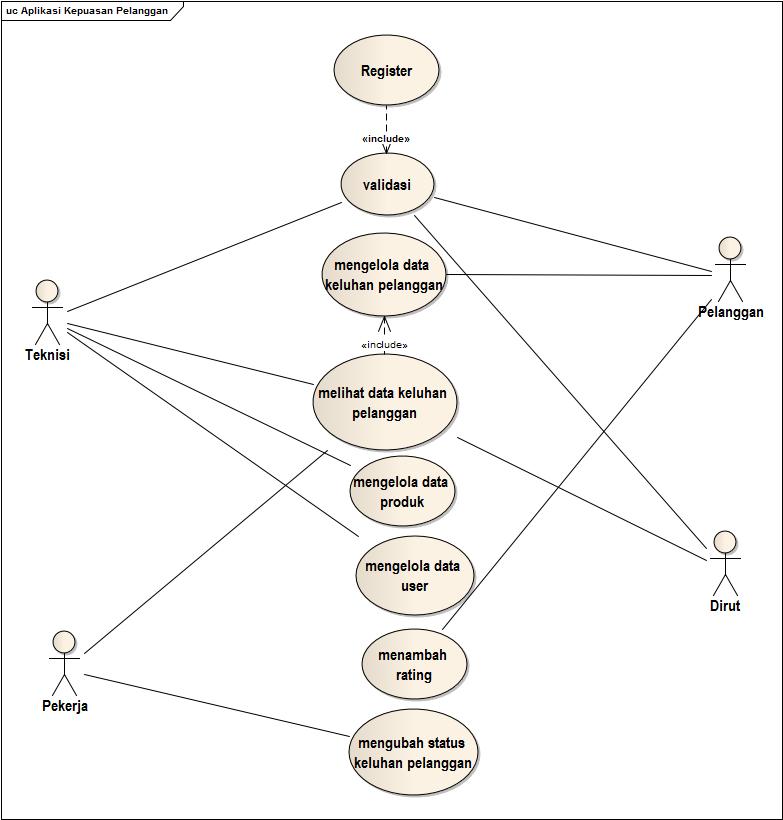
1. Pelanggan menyampaikan keluhan langsung kepada pekerja dengan datang ke *Software House* Lampung.
2. Pekerja menanggapi keluhan pelanggan tersebut.
3. Pekerja menyampaikan kepada pelanggan bahwa keluhannya telah dikerjakan melalui telepon.
4. Pelanggan mendapat informasi bahwa keluhannya telah dikerjakan 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)
2. Melihat data keluhan pelanggan.
3. Mengelola data produk.
4. Melihat dan mengubah data *user* atau pelanggan.
5. Level Dirut
6. Melihat data keluhan pelanggan.
7. Level *user* atau pelanggan
8. Mengelola data keluhan pelanggan.
9. Menambah rating.
10. Level Pekerja
11. Melihat data keluhan pelanggan.
12. Mengubah status keluhan pelanggan.
    * 1. **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.

1. *Use case diagram* level admin

*Use case diagram* level admin menggambarkan kelakukan 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.

1. *Use case diagram* level pelanggan

*Use case diagram* level pelanggan menggambarkan kelakukan 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.

1. *Use case diagram* level pekerja

*Use case diagram* level pekerja menggambarkan kelakukan 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.

1. 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*

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

1. 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

|  |  |  |
| --- | --- | --- |
| *Use case name* | Validasi | |
| *Actor* | Admin, Pelanggan, dan Pekerja | |
| *Description* | *Actor* menggunakan *use case* ini untuk melakukan *login* ke dalam sistem. | |
| *Precondition* | *Actor* belum berhasil melakukan validasi | |
| *Pastcondition* | *Actor* berhasil melakukan validasi | |
| *Basic flow* | *Actor* | *System* |
| 1. Membuka Aplikasi   3. Mengisi *form login* | 1. Menampilkan aplikasi   4. Cek validasi *email mm*dan *password*  5. Jika *email* dan mm*password* benar, mmsistem akan mmmenampilkan mmhalaman *home*  6. Jika *email* dan mm*password* salah, mmmaka proses akan mmkembali ke no. 3 |

1. 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

|  |  |  |
| --- | --- | --- |
| *Use case name* | Melihat data keluhan pelanggan | |
| *Actor* | Admin dan Pekerja | |
| *Description* | *Actor* menggunakan *use case* ini untuk dapat melihat data keluhan pelanggandi dalam sistem. | |
| *Precondition* | Admin belum bisa melihat data keluhan pelanggan | |
| *Pastcondition* | Admin berhasil melihat data keluhan pelanggan | |
| *Basic flow* | *Actor* | *System* |
| Tabel 7. (Lanjutan) |  |  |
|  | 1. Membuka aplikasi | 1. Menampilkan aplikasi |
| 1. Melakukan sesuai skenario pada tabel *login* | 1. Menampilkan halaman utama (*home*) |
| 1. Melihat data keluhan pelanggan | 1. Menampilkan halaman data keluhan pelanggan |

1. 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

|  |  |  |
| --- | --- | --- |
| *Use case name* | Mengelola data produk | |
| *Actor* | Admin | |
| *Description* | *Actor* menggunakan *use case* ini untuk dapat mengelola data produk di dalam sistem yaitu melihat, menambah, mengubah dan menghapus. | |
| *Precondition* | Admin belum bisa mengelola data produk | |
| *Pastcondition* | Admin berhasil mengelola data produk | |
| *Basic flow* | *Actor* | *System* |
|  | 1. Membuka aplikasi | 1. Menampilkan aplikasi |
| 1. Melakukan sesuai skenario pada tabel *login* | 1. Menampilkan halaman utama (*home*) |
| 1. Melihat data produk | 1. Menampilkan halaman data produk |
| 1. Menambah data produk | 1. Menampilkan *form* tambah produk |
| 1. Mengubah data produk | 1. Menampilkan *form* *edit* produk |
| 1. Menghapus data produk | 1. Menampilkan *form* hapus produk |

1. 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*

|  |  |  |
| --- | --- | --- |
| *Use case name* | Mengelola data *user* | |
| *Actor* | Admin | |
| *Description* | *Actor* menggunakan *use case* ini untuk dapat mengelola data *user* di dalam sistem yaitu melihat, mengubah dan menghapus. | |
| *Precondition* | Admin belum bisa mengelola data *user* | |
| *Pastcondition* | Admin berhasil mengelola data *user* | |
| *Basic flow* | *Actor* | *System* |
|  | 1. Membuka aplikasi | 1. Menampilkan aplikasi |
| 1. Melakukan sesuai skenario pada tabel *login* | 1. Menampilkan halaman utama (*home*) |
| 1. Melihat data *user* | 1. Menampilkan halaman data *user* |
| 1. Mengubah data *user* | 1. Menampilkan *form* *edit* *user* |
| 1. Menghapus data *user* | 1. Menampilkan *form* hapus *user* |

1. Deksripsi *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

|  |  |  |
| --- | --- | --- |
| *Use case name* | Mengelola data keluhan pelanggan | |
| *Actor* | Pelanggan | |
| *Description* | *Actor* menggunakan *use case* ini untuk dapat mengelola data keluhan pelanggan di dalam sistem yaitu melihat, menambah, mengubah dan menghapus. | |
| *Precondition* | Pelanggan belum bisa mengelola data keluhan pelanggan | |
| *Pastcondition* | Pelanggan berhasil mengelola data keluhan pelanggan | |
| *Basic flow* | *Actor* | *System* |
|  | 1. Membuka aplikasi | 1. Menampilkan aplikasi |
| 1. Melakukan sesuai skenario pada tabel *login* | 1. Menampilkan halaman utama (*home*) |
| 1. Melihat data keluhan pelanggan | 1. Menampilkan halaman data keluhan pelanggan |
| Tabel 10. (Lanjutan) |  | |
|  | 1. Menambah data keluhan pelanggan | 1. Menampilkan *form* tambah keluhan pelanggan |
| 1. Mengubah data keluhan pelanggan | 1. Menampilkan *form* *edit* keluhan pelanggan |
| 1. Menghapus data keluhan pelanggan | 1. Menampilkan *form* hapus keluhan pelanggan |

1. 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*

|  |  |  |
| --- | --- | --- |
| *Use case name* | Menambah *rating* | |
| *Actor* | Pelanggan | |
| *Description* | *Actor* menggunakan *use case* ini untuk dapat menambah *rating* kepuasan pelanggan di dalam sistem | |
| *Precondition* | Pelanggan belum bisa menambah *rating* | |
| *Pastcondition* | Pelanggan berhasil menambah *rating* | |
| *Basic flow* | *Actor* | *System* |
|  | 1. Membuka aplikasi | 1. Menampilkan aplikasi |
| 1. Melakukan sesuai skenario pada tabel *login* | 1. Menampilkan halaman utama (*home*) |
| 1. Melihat data keluhan pelanggan | 1. Menampilkan halaman data keluhan pelanggan |
| 1. Menambah *rating* kepuasan pelanggan | 1. Menampilkan halaman data keluhan pelanggan 2. Jika sudah menambah *rating*, maka akan menampilkan halaman data keluhan pelanggan yang sudah di*rating* |

1. Deskripsi *use case* mengubah status keluhan pelanggan

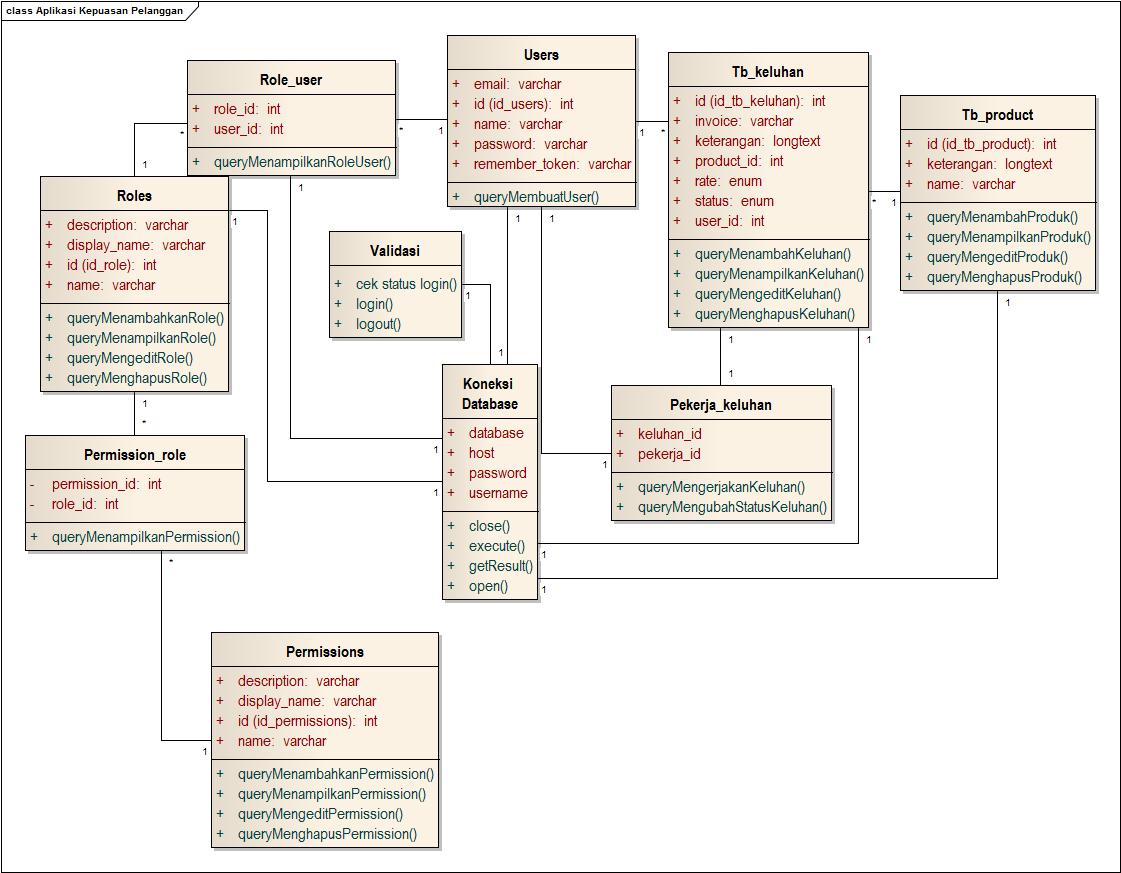
Deskripsi *use case* mengubah status keluhan pelanggan merupakan penjabaran tentang alur dari proses mengubah status keluhan pelanggan. Deskripsi *use case* mengubah status keluhan pelanggandisajikan pada tabel 12.

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

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

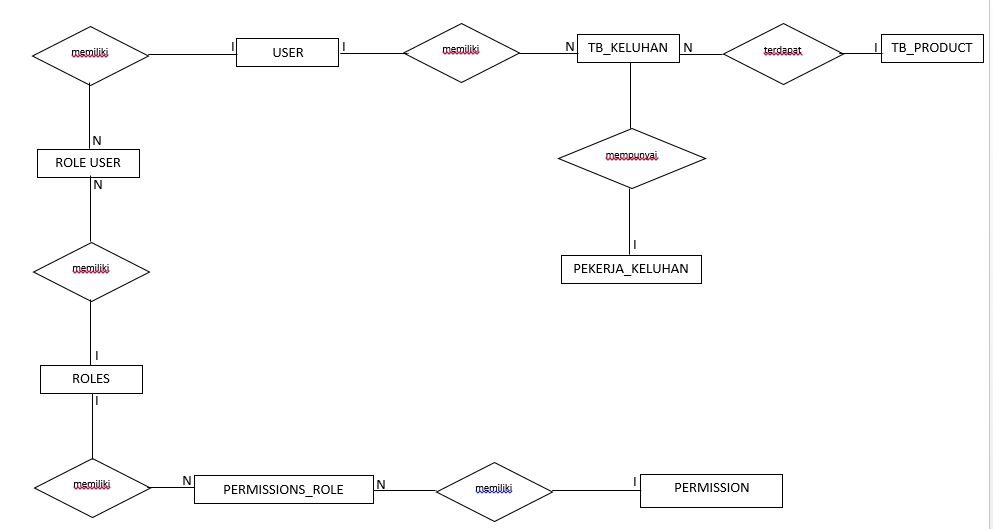
* + - 1. **Desain *class diagram***

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

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Id | *Integer* | 10 | - |
| Name | *Varchar* | 30 | - |
| Email | *Varchar* | 25 | - |
| Password | *Varchar* | 10 | - |
| Remember\_token | *Varchar* | 100 | - |

1. *Database* : laravel\_admin.sql

Tabel : tb\_keluhan

*Primary Key* : id

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Id | *Integer* | 10 | - |
| Invoice | *Varchar* | 10 | - |
| User\_id | *Integer* | 10 | - |
| Product\_id | *Integer* | 10 | - |
| Keterangan | *Longtext* | - | - |
| Status | *Enum* | - | - |
| Rate | *Enum* | - | - |

1. *Database* : laravel\_admin.sql

Tabel : tb\_product

*Primary Key* : id

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Id | *Integer* | 10 | - |
| Name | *Varchar* | 30 | - |
| Keterangan | *Longtext* | - | - |

1. *Database* : laravel\_admin.sql

Tabel : pekerja\_keluhan

*Primary Key* : -

*Foreign Key* : keluhan\_id

pekerja\_id

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Keluhan\_id | *Integer* | 10 | - |
| Pekerja\_id | *integer* | 10 | - |

1. *Database* : laravel\_admin.sql

Tabel : *roles*

*Primary Key* : id

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Id | *Integer* | 10 | - |
| Name | *Varchar* | 30 | - |
| Display\_name | *Varchar* | 30 | - |
| Description | *Varchar* | 30 | - |

1. *Database* : laravel\_admin.sql

Tabel : *role\_user*

*Primary Key* : -

*Foreign Key* : *user\_id*

*role\_id*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| User\_id | *Integer* | 10 | - |
| Role\_id | *Integer* | 10 | - |

1. *Database* : laravel\_admin.sql

Tabel : *permissions*

*Primary Key* : id

|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Id | *Integer* | 10 | - |
| Name | *Varchar* | 30 | - |
| Display\_name | *Varchar* | 30 | - |
| Description | *Varchar* | 30 | - |

1. *Database* : laravel\_admin.sql

Tabel : *permission\_role*

*Primary Key* : -

*Foreign Key* : *permission\_id*

*role\_id*

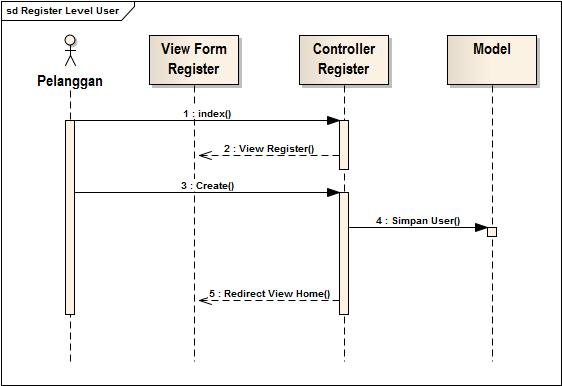
|  |  |  |  |
| --- | --- | --- | --- |
| ***Field Name*** | ***Data Type*** | ***Length*** | ***Ket*** |
| Permission\_id | *Integer* | 10 | - |
| Role\_id | *Integer* | 10 | - |

* + - 1. **Desain *sequence diagram***

*Sequence diagram* adalah diagram yang digunakan untuk mendefinisikan *input* dan *output* serta urutan interkasi 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 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*

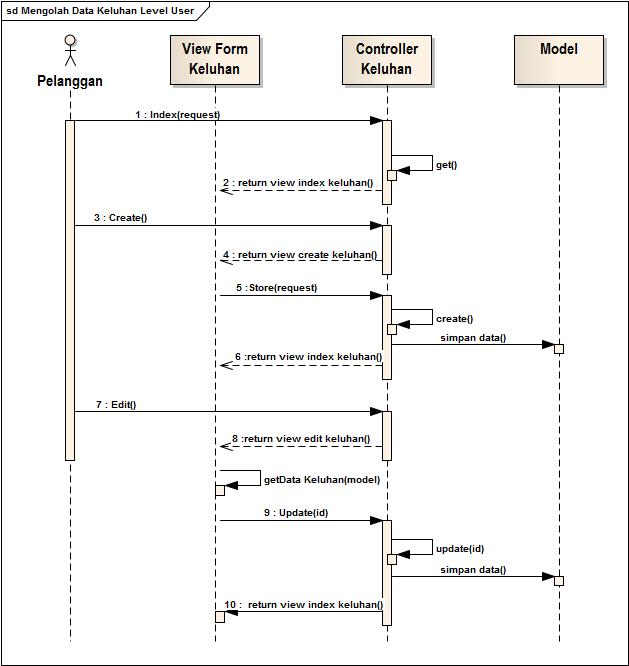
*Sequence diagram* untuk melakukan registrasi pendaftaran pelanggan menggambarkan alur proses di dalam pembuatan *member*. Rancangan *sequence diagram* melakukan registrasi pelanggan disajikan pada Gambar 8.



Gambar 8. *Sequence diagram* melakukan registrasi pelanggan

1. *Sequence diagram* mengelola data keluhan pelanggan

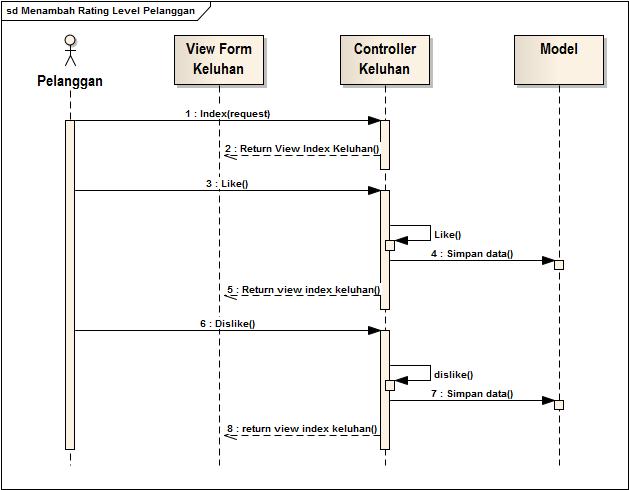
*Sequence diagram* mengelola data keluhan pelanggan menggambarkan alur proses dalam menambah, melihat, mengubah dan menghapus data keluhan pelanggan. Rancangan *sequence diagram* mengelola data keluhan pelanggan disajikan pada Gambar 9.



Gambar 9. *Sequence diagram* mengelola data keluhan pelanggan

1. *Sequence diagram* menambah *rating*

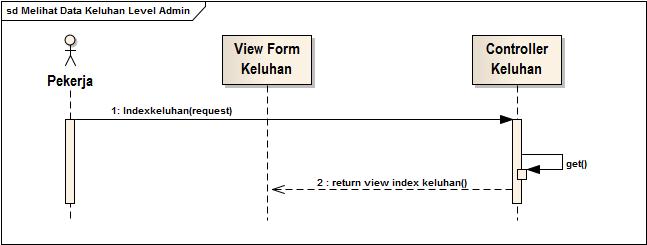
*Sequence diagram* menambah *rating* menggambarkan alur proses pelanggan menambah *rating*  keluhan pelanggan. Rancangan *sequence diagram* menambah *rating* disajikan pada Gambar 10.



Gambar 10. *Sequence diagram* menambah *rating*

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

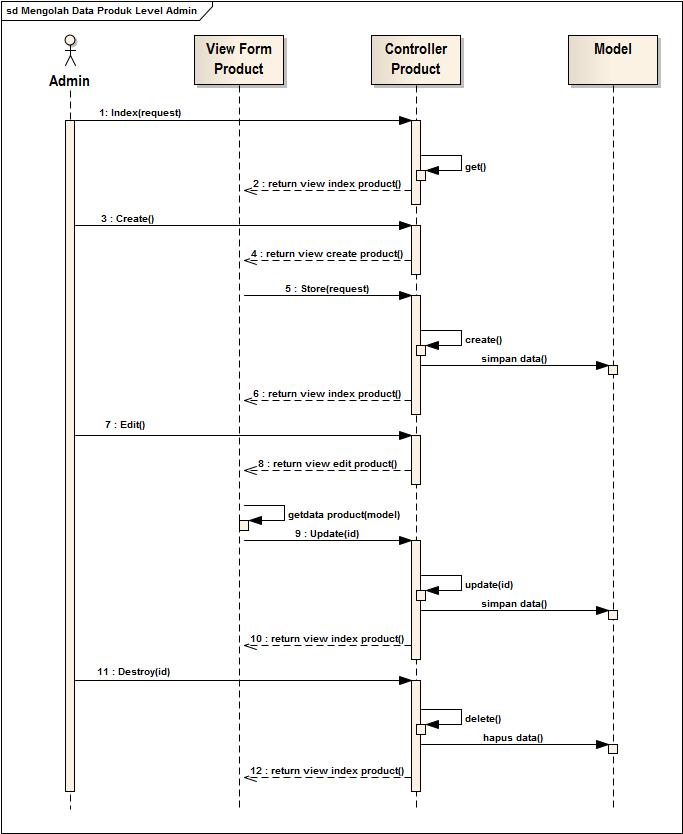
*Sequence diagram* melihat data keluhan pelanggan level admin menggambarkan alur proses dalam melihat data keluhan pelanggan pada aplikasi program. Rancangan *sequence diagram* melihat data keluhat pelanggan level admin disajikan pada Gambar 11.



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

1. *Sequence diagram* mengelola data produk level admin

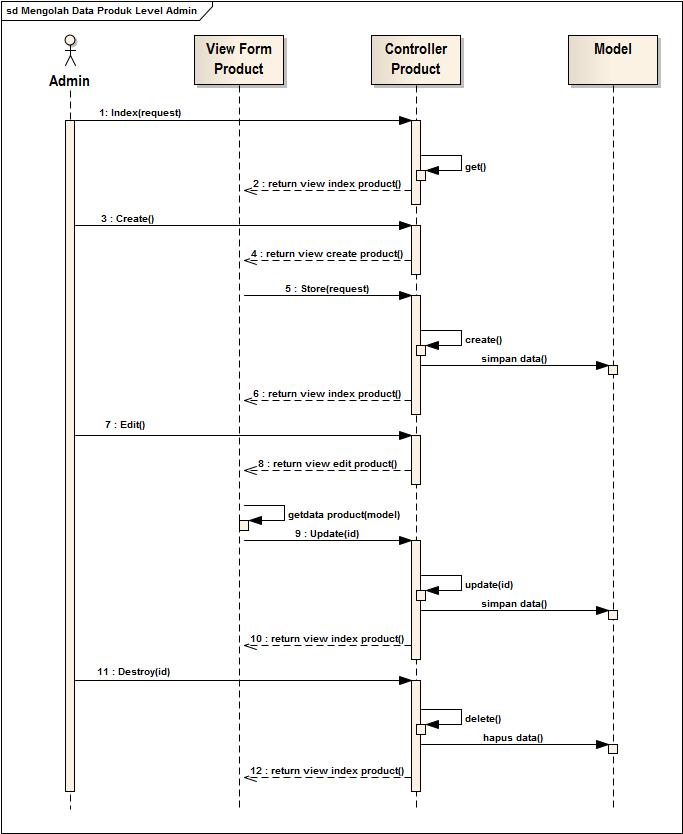
*Sequence diagram* mengelola data produk level admin menggambarkan alur proses dalam menambah, melihat, mengubah dan menghapus data produk. Rancangan *sequence diagram* mengelola data produk disajikan pada Gambar 12.



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

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

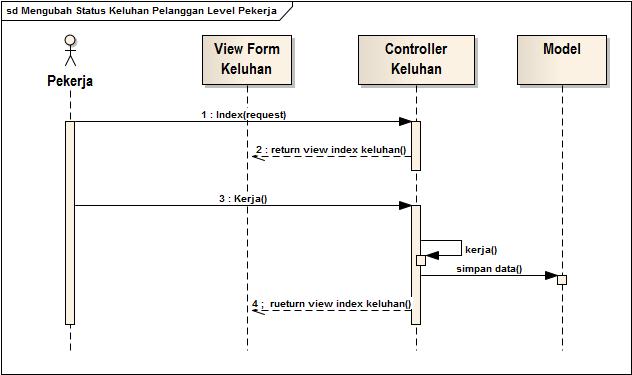
*Sequence diagram* mengelola data *user* level admin menggambarkan alur proses dalam melihat, mengubah dan menghapus data *user*. Rancangan *sequence diagram* mengelola data *user* disajikan pada Gambar 13.



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

1. *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.



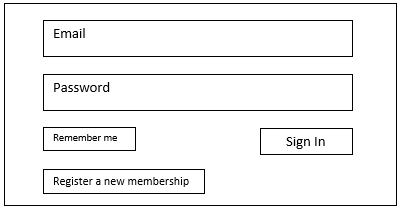
Gambar 14. *Sequence diagram* mengubah status keluhan pelanggan level admin

* + - 1. **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*

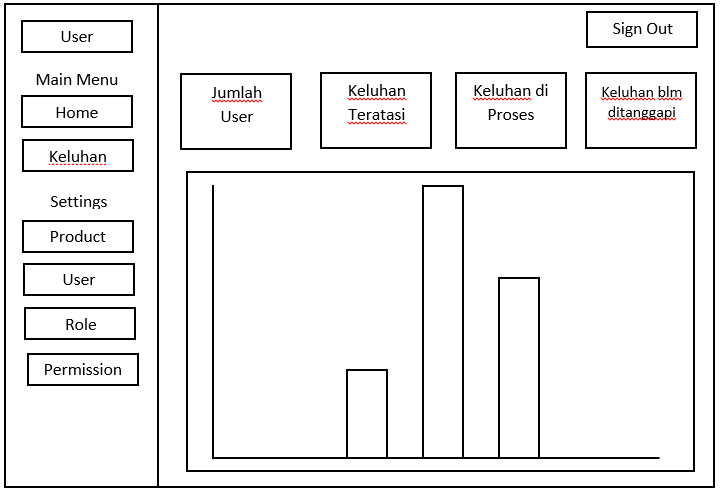
*Form* *login* terdiri dari kolom *username* dan *password* untuk akses masuk kedalam aplikasi, apabila *username* dan *password* benar dan sesuai dengan *database* maka *user* dapat masuk ke dalam aplikasi sesuai hak aksesnya. Rancangan tampilan *form login* disajikan pada Gambar 15.



Gambar 15. Rancangan tampilan *form login*

1. Rancangan tampilan halaman utama admin

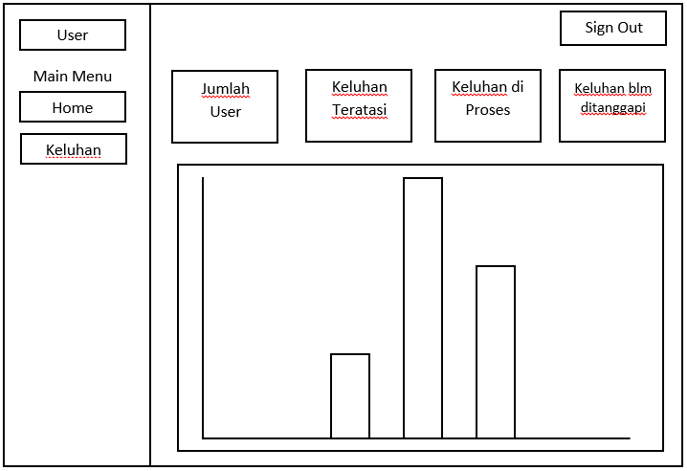
Halaman utama untuk *admin* yang terdiri dari beberapa menu yakni *user*, *home*, keluhan, *product*, *role*, *permission*, grafik, dan *logout*. Rancangan tampilan menu utama *admin* disajikan pada Gambar 16.



Gambar 16. Rancangan tampilan halaman utama admin

1. Rancangan tampilan halaman utama pelanggan

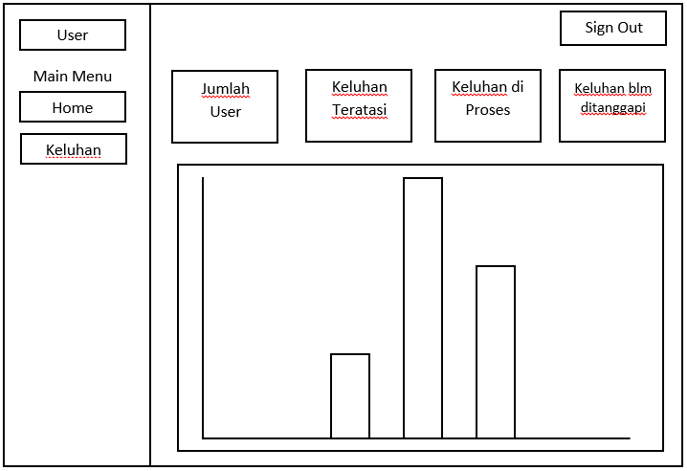
Halaman utama untuk pelanggan terdiri dari beberapa menu yakni *home*, keluhan, grafik, dan *logout*. Rancangan tampilan menu utama pelanggan disajikan pada Gambar 17.



Gambar 17. Rancangan tampilan halaman utama pelanggan

1. Rancangan tampilan halaman utama pekerja

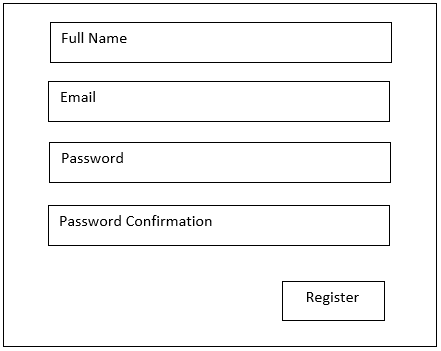
Halaman utama untuk pekerja terdiri dari beberapa menu yakni *home*, keluhan, grafik, dan *logout*. Rancangan tampilan menu utama pekerja disajikan pada Gambar 18.



Gambar 18. Rancangan tampilan halaman utama pekerja

1. Rancangan *form register*

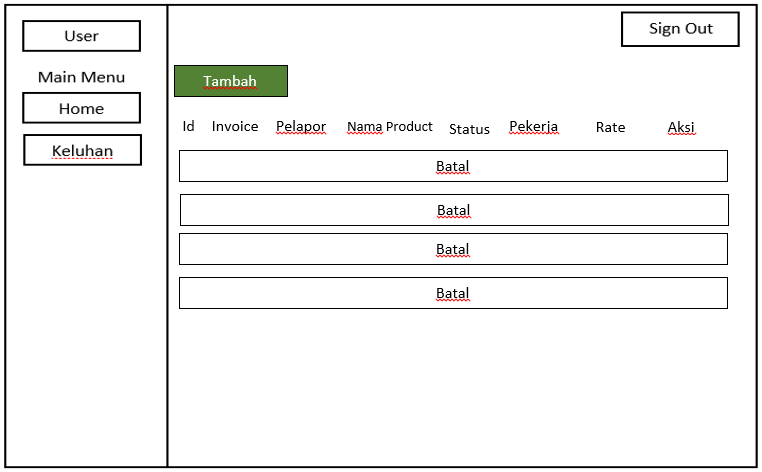
*Form register* adalah adalah tampilan halaman untuk mendaftar atau membuat akun di halaman registrasi pada *website* aplikasi pelaporan keluhan pelanggan di *Software House* Lampung. Rancangan halaman *register* disajikan pada Gambar 19.



Gambar 19. Rancangan tampilan *form*  *register*

1. Rancangan tampilan menu keluhan pelanggan

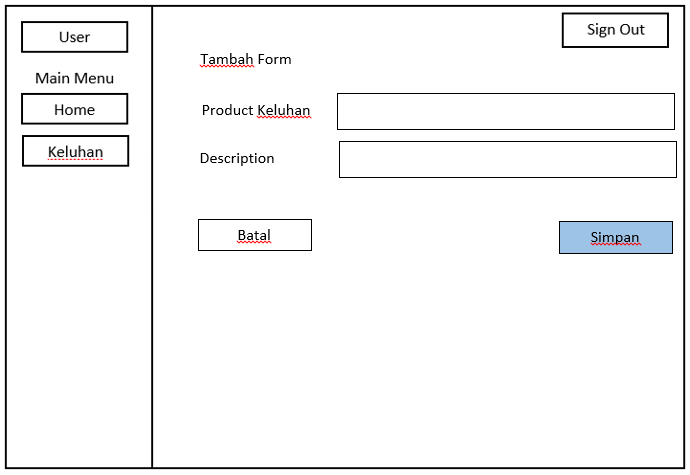
Tampilan menu keluhan pelanggan ini terdapat beberapa aksi seperti menambah, mengubah, dan menghapus. Rancangan tampilan menu keluhan pelanggan disajikan pada Gambar 20.



Gambar 20. Rancangan tampilan menu keluhan pelanggan

1. Rancangan tampilan *form* tambah keluhan pelanggan

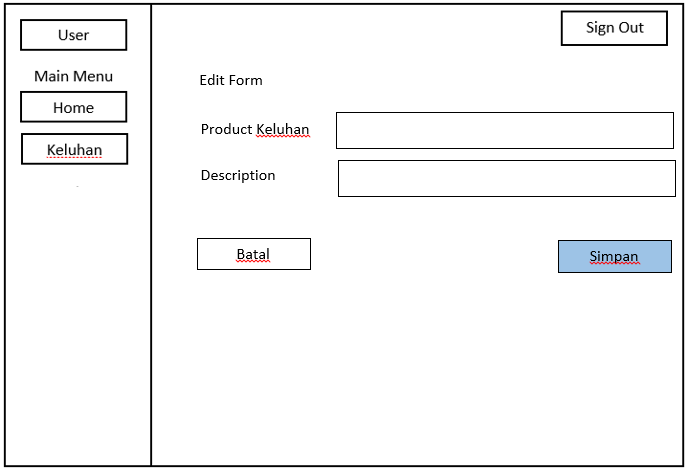
*Form* tambah adalah tampilan halaman untuk menambah keluhan pelanggan di halaman *form*  tambah pada *website* aplikasi pelaporan keluhan pelanggan di *Software House* Lampung. Rancangan tampilan *form* tambah keluhan pelanggan disajikan pada Gambar 21.



Gambar 21. Rancangan tampilan *form* tambah keluhan pelanggan

1. Rancangan tampilan *form edit* keluhan pelanggan

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



Gambar 22. Rancangan tampilan *form edit* keluhan pelanggan

* + 1. **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 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')) {  $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();  }  } else {  if (Auth::user()->hasRole('pelanggan')) {  $cks = Keluhan::where('status', 3)->where('user\_id', Auth::user()->id)->get();  $ckp = Keluhan::where('status', 2)->where('user\_id', Auth::user()->id)->get();  $ckb = Keluhan::where('status', 1)->where('user\_id', Auth::user()->id)->get();  for ($i = 1; $i <= 12; $i++) {  $chks[$i] = Keluhan::where('status', 3)  ->where('user\_id', Auth::user()->id)  ->whereMonth('created\_at', '=', $i)  ->whereYear('created\_at', '=', date('Y'))->count();  $chkp[$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'));  }} |

1. Penulisan kode program pada Keluhan*Controller*.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');  }} |

1. 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();  }} |

1. 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');  }  } |

1. 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');  }} |

1. Penulisan kode program pada *UserController*.php

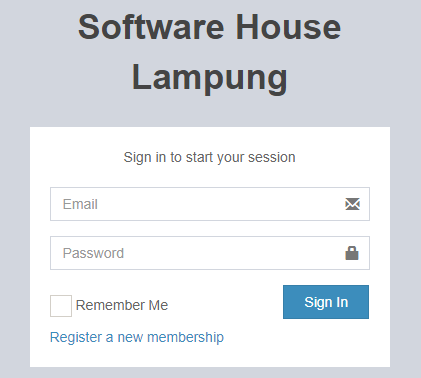
|  |
| --- |
| <?php  namespace App\Http\Controllers;  use App\Role;  use App\RoleUser;  use App\User;  use Illuminate\Http\Request;  class UserController extends Controller  {  public function index()  {  $user = User::all();  $roles = Role::all();  return view('user.index', compact('user', 'roles'));  }  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;  $ru->save();  }  public function deleUserRole($user, $role)  {  $ru = RoleUser::where([  ['user\_id', '=', $user],  ['role\_id', '=', $role]]);  $ru->delete();  }} |

* + - 1. **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*

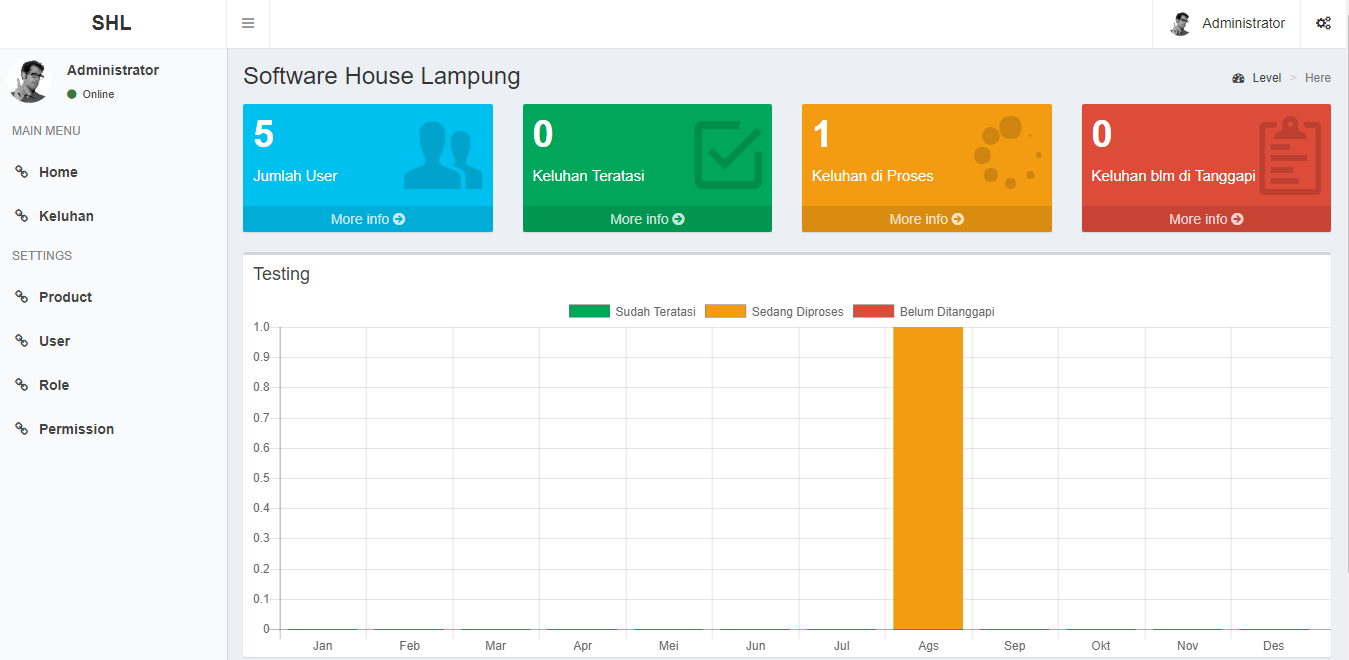
*Form login* berfungsi untuk mendapatkan hak akses, untuk dapat mengakses aplikasi tersebut admin, pelanggan dan pekerja harus memasukkan *email* dan *password*. Tampilan *form login* disajikan pada Gambar 23.



Gambar 23. Tampilan *form login*

1. Tampilan halaman utama admin

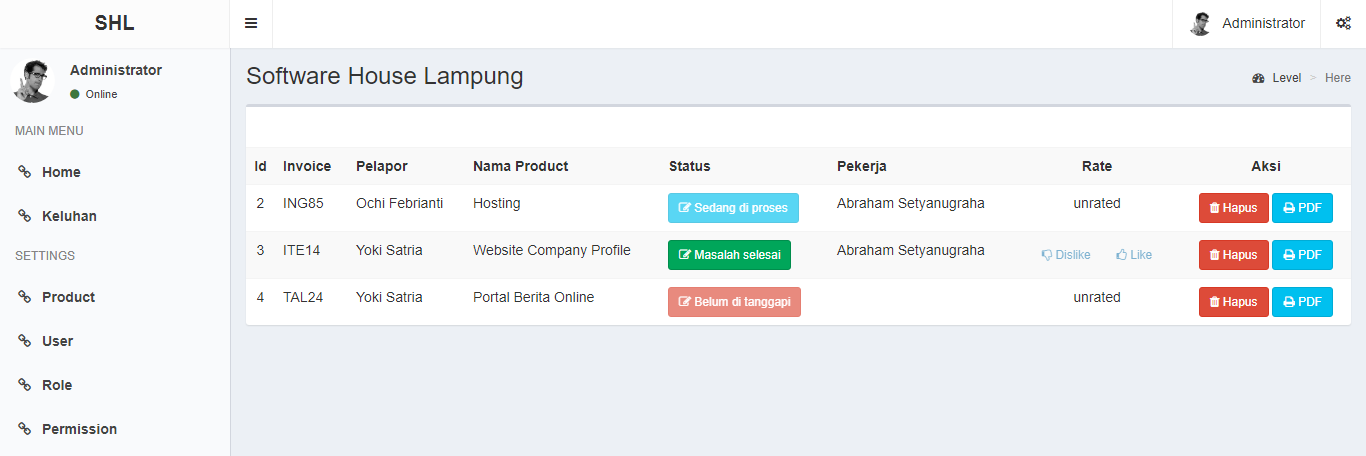
Halaman utama admin menampilkan grafik keluhan pelanggan, dan menampilkan menu yang dapat dilakukan oleh admin yaitu menu *home*, keluhan, *user*, *product*, *role*, *permission*. Tampilan halaman utama admin disajikan pada Gambar 24.



Gambar 24. Tampilan halaman utama admin

1. Tampilan halaman menu keluhan pelanggan level admin

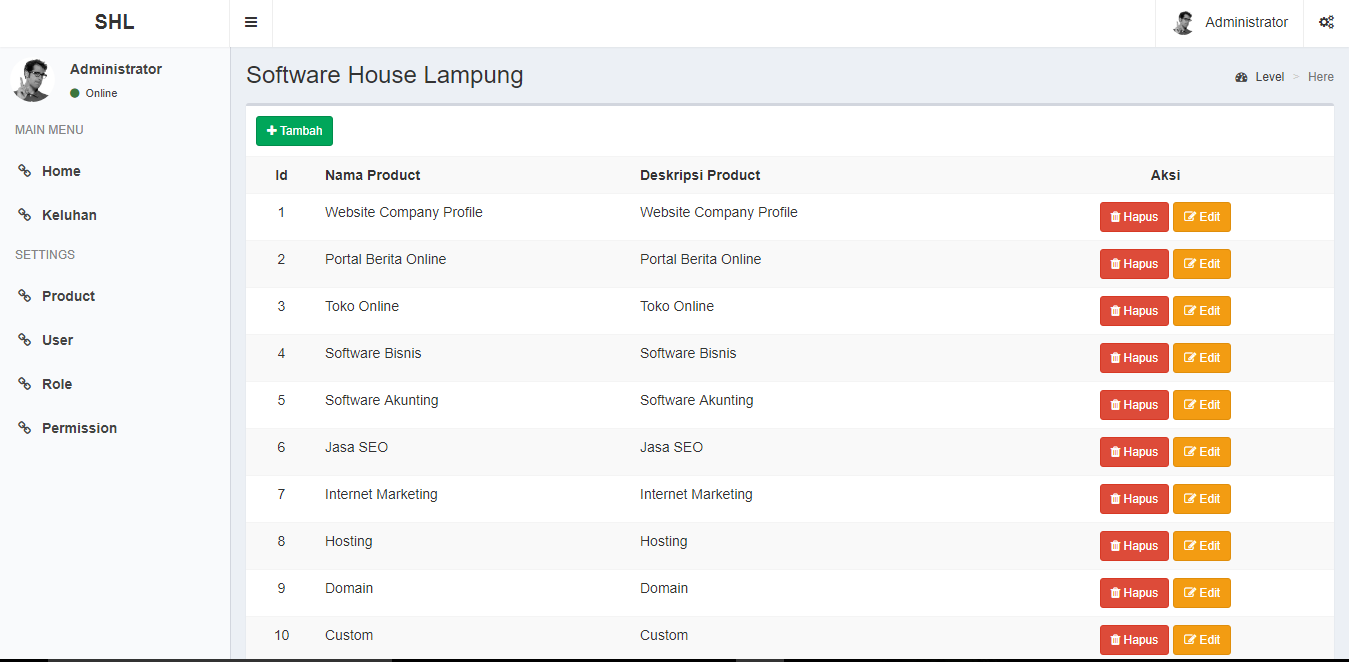
Halaman menu keluhan pelanggan pada level admin berfungsi untuk menampilkan data keluhan pelanggan dengan dapat menghapus dan mencetak laporan keluhan pelanggan. Tampilan halaman menu keluhan pelanggan disajikan pada Gambar 25.



Gambar 25. Tampilan halaman menu keluhan pelanggan

1. Tampilan halaman menu *product* level admin

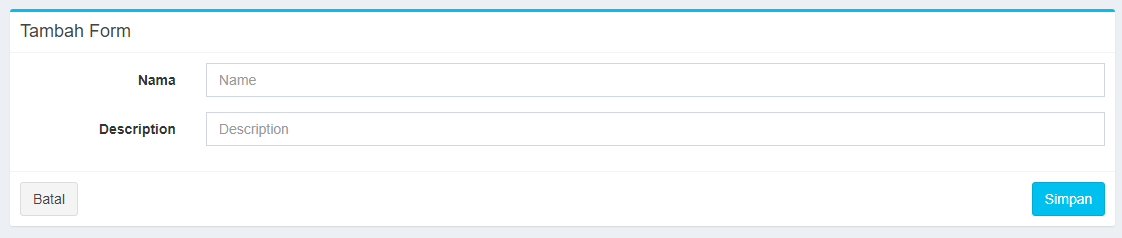
Halaman menu *product* pada level admin berfungsi untuk menampilkan data *product* dengan dapat menambah, mengedit dan menghapus data *product*. Tampilan halaman menu *product* disajikan pada Gambar 26.



Gambar 26. Tampilan halaman menu *product*

1. Tampilan *form* tambah *product*

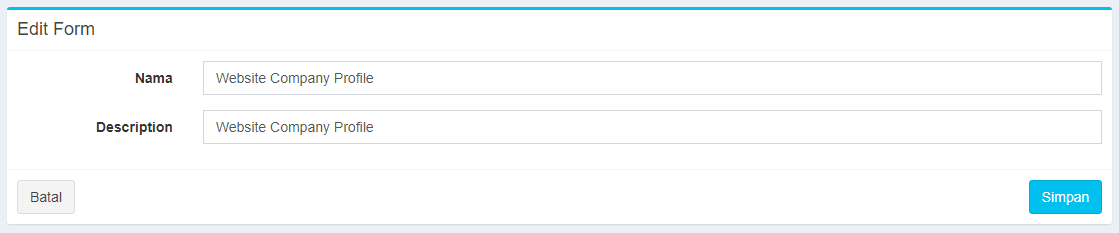
Tampilan *form* tambah *product* digunakan untuk menambah data *product*. Tampilan *form* tambah *product* disajikan pada Gambar 27.



Gambar 27. Tampilan *form* tambah *product*

1. Tampilan *form* *edit product*

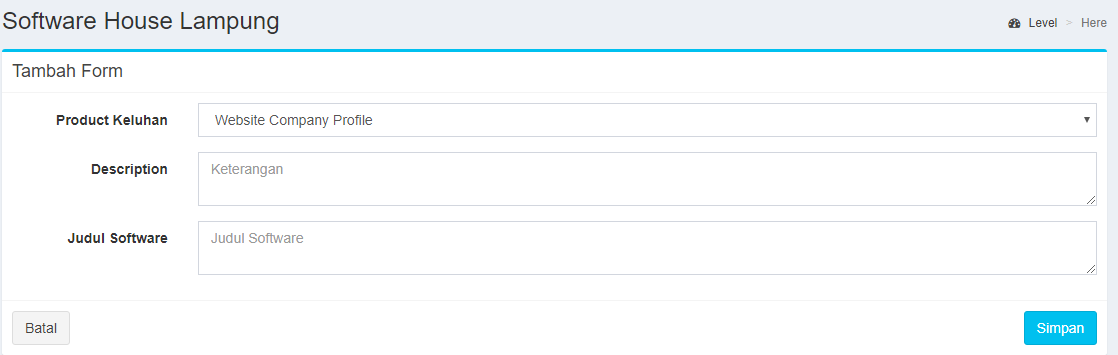
Tampilan *form edit product* digunakan untuk mengubah data *product*. Tampilan *form edit product* disajikan pada Gambar 28.



Gambar 28. Tampilan *form edit product*

1. Tampilan *form* tambah keluhan pelanggan level pelanggan

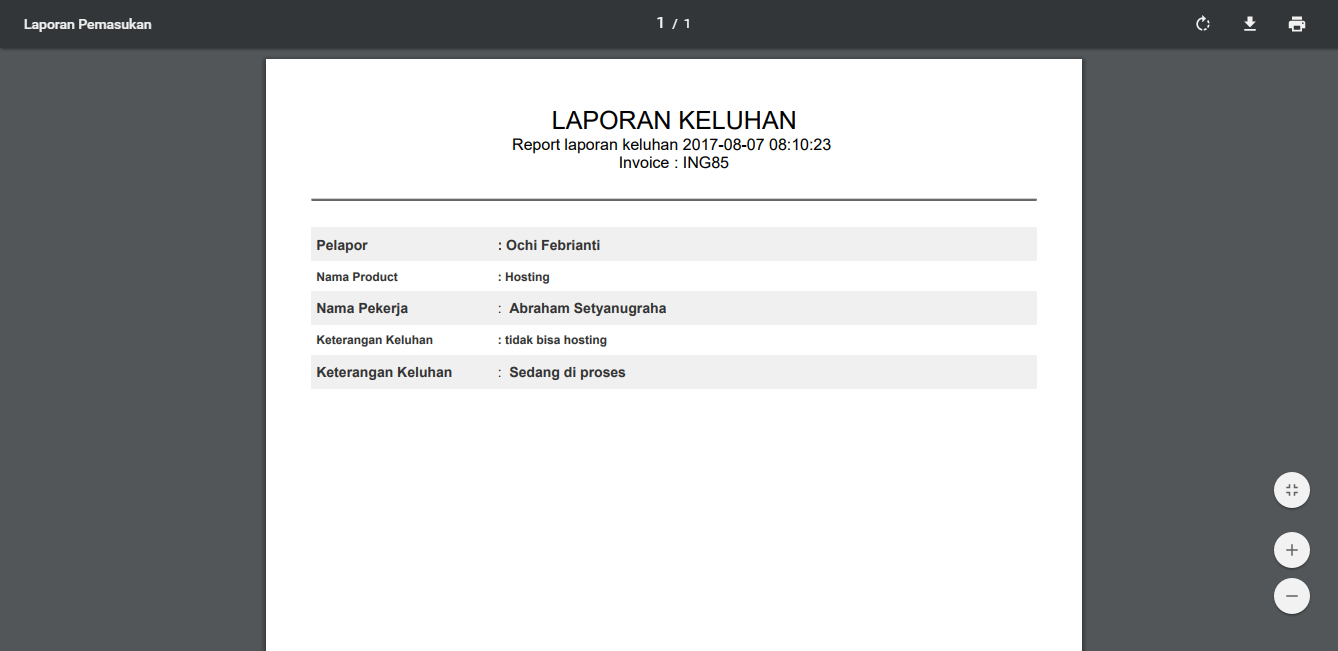
Tampilan *form* tambah keluhan pelanggan digunakan untuk menambah data keluhan pelanggan. Tampilan *form* tambah keluhan pelanggan disajikan pada Gambar 29.



Gambar 29. Tampilan *form* tambah keluhan pelanggan

1. 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

* + 1. **Pengujian Sistem**

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

**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.

* + - 1. **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.

* + 1. **Pendukung Sistem**

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