BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pembahasan pada bab-bab sebelumnya, maka dapat ditarik beberapa kesimpulan yaitu :

1. Pembangunan perangkat lunak dengan menggunakan konsep *Object Relational Database Management System* (ORDBMS) merupakan hal yang belum banyak digunakan dikarenakan konsep ORDBMS masih cukup baru. Penggunaan fitur-fitur seperti *object type, nested table* dimaksudkan untuk melakukan explorasi terhadap kemampuan ORDBMS. Penggunaan *nested table* dapat mengurangi frekuensi relasi antar tabel yang terjadi.
2. Walaupun masih dalam taraf sederhana, namun aplikasi *Customer Relationship Management* (CRM) ini berhasil dibangun sesuai dengan tujuan semula adalah membangun sebuah perangkat lunak yang mampu membantu dalam melakukan pengelolaan atau manajemen untuk membina hubungan yang baik dengan klien/*customer*, yang meliputi pencatatan, penyimpanan, pencarian, serta memberikan laporan yang aktual dan informatif terhadap data-data yang berhubungan dengan relasi antara perusahaan dan *customer*. Manajemen tersebut dapat didukung dengan adanya fasilitas mail untuk menghubungkan *customer* dengan pegawai yang berkepentingan serta menyediakan laporan yang aktual berkaitan dengan pembayaran premi bulanan yang menjadi kewajiban *customer*. Di samping fasilitas penghubung antara

customer dengan pegawai, CRM juga menyediakan fasilitas interaksi antara pegawai demi menunjang proses manajemen terhadap hubungan dengan *customer*.

5.2. Saran

Beberapa saran yang dapat diambil dari proses analisis sampai pada pembuatan tugas akhir ini adalah sebagai berikut :

1. Perangkat lunak diharapkan dapat dikembangkan lebih lanjut dengan mengintegrasikan CRM dengan *data warehouse* yang memiliki informasi mengenai profil dan sejarah *customer*, sehingga proses *retain* dan *extend* akan lebih efektif.
2. Perangkat lunak diharapkan dapat dikembangkan lebih lanjut dengan penambahan fungsionalitas dan interface yang lebih menarik untuk pelayanan agar *customer* merasa lebih nyaman dan terikat dengan fasilitas yang diberikan.

DAFTAR PUSTAKA

Connolly, Thomas dan Begg, C., 2005, *Database Systems Fourth Edition*, Pearson Education Limited, London.

Findlay, Cliff, 2000, *CRM. What's it all about*. CRM Consultant at Latitude Solutions Ltd, ITToolbox Portal for CRM.

Harper, Simon, 2000, *Customer Relationship Management (CRM): Well what is it really?* NHANZ Ltd., ITToolbox Portal for CRM.

http://en.wikipedia.org/wiki/Business intelligence

http://en.wikipedia.org/wiki/Customer relationship management

Khera, Mandeep, 2000, *Customer Relationship Management – Beyond the "buzz"*, United Customer Management Solutions, ITToolbox Portal for CRM.

Mallach, Efren, 2000, *Decision Support and Data Warehouse System*. Mc Graw Hill, New York.

NIIT, 2001, *Oracle 9i Object Relational Database Management Systems*. NIIT Online Learning Ltd.

Nugroho, Adi, 2005, *Rational Rose untuk Pemrograman Berorientasi Objek*. Informatika Bandung.

Sambasivan, TN., 2000, *CRM for Dummies*. ITToolbox Portal for CRM.

Turban, Efraim et.al., 2001, *Information Technology for Management Third Edition*, John Wiley & Sons, Inc, New York.

Turban, Efraim et.al., 2001, *Introduction to Information Technology*, John Wiley & Sons, Inc., New York.

Warta JWC, "Vol. 8/IV/Agustus/2006". 2006. Universitas Bina Nusantara Indonesia.

Wisnubhadra, I., 2005, *Bahan Kuliah Basis Data dan Basis Data Lanjut*. Universitas Atma Jaya Yogyakarta.

LAMPIRAN

SKPL

SPESIFIKASI KEBUTUHAN PERANGKAT LUNAK

APLIKASI *CRM* (*CUSTOMER RELATIONSHIP MANAGEMENT*) PADA PERUSAHAAN ASURANSI DENGAN *OBJECT RELATIONAL DATABASE MANAGEMENT SYSTEM*

Dipersiapkan oleh:

ANGGISESA JALULAGA / 3712

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	

INDEX TGL	22 Feb 2007	30 Mei 2007					
Ditulis oleh	AJ	AJ					
Diperiksa oleh	IW dan YSP	IW dan YSP					
Disetujui oleh	IW dan YSP	IW dan YSP					

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

1. Pendahuluan

1.1. Tujuan

Tujuan dari dokumen spesifikasi kebutuhan perangkat lunak (SKPL) dalam pengembangan perangkat lunak *CRM* (*Customer Relationship Management*) yaitu mendefinisikan spesifikasi dan kebutuhan sebagai tahap awal pengembangan perangkat lunak. Aplikasi *CRM* sendiri nantinya merupakan perangkat lunak yang digunakan untuk melakukan manajemen dan digunakan untuk membantu dalam proses pengambilan keputusan bisnis yang strategis, terutama dalam kaitannya untuk menjaga hubungan yang baik antara pihak perusahaan dengan pihak klien/*customer*.

Secara lebih lanjut, SKPL juga merupakan suatu bentuk alat yang digunakan oleh pihak *developer* (pengembang) dan *user* (pengguna) untuk berkomunikasi demi tercapainya suatu pemahaman yang sama terhadap penyusun dasar dari sebuah perangkat lunak yang akan dikembangkan. Hal ini akan mempermudah dalam pembelajaran dan pengembangan terhadap perangkat lunak (*Software*) yang bersangkutan. Di dalam SKPL ini akan dibahas mengenai antarmuka eksternal (antarmuka antara sistem dengan sistem lain perangkat lunak dan perangkat keras, dan pengguna) perfomansi (kemampuan perangkat lunak dari segi kecepatan, tempat penyimpanan yang dibutuhkan, serta keakuratan), atribut (*feature-feature* tambahan yang dimiliki sistem), mendefinisikan fungsi perangkat

lunak, serta mendefinisikan batasan perancangan perangkat lunak.

1.2. Lingkup Masalah

Perangkat Lunak *Customer Relationship Management* ini merupakan perangkat lunak yang mengambil *business process* dari PT. Asuransi Allianz Life Indonesia dengan tujuan membangun sebuah perangkat lunak yang mampu membantu dalam melakukan pengelolaan atau manajemen untuk membina hubungan yang baik dengan klien/*customer*, yang meliputi pencatatan, penyimpanan, pencarian, serta memberikan laporan yang aktual dan informatif terhadap data-data yang berhubungan dengan relasi antara perusahaan dan *customer*. Data-data tersebut meliputi:

- Data user, merupakan pengguna dari perangkat lunak *CRM* ini.
- Data produk asuransi, merupakan jenis-jenis fasilitas asuransi yang ditawarkan.
- Data *customer*, merupakan pengguna atau anggota dari produk asuransi yang ditawarkan.
- Data *employee*, merupakan pegawai pada perusahaan. *Employee* nantinya akan dibagi menjadi 3 jenis, yaitu:
  - *Customer care*, merupakan pihak yang menjaga hubungan antara perusahaan dengan *customer* dan juga bertindak sebagai *Branch Manager* pada perusahaan yang bersangkutan.

- Agen (marketing/sales), merupakan pihak yang bertugas mencari *customer* dan merupakan bawahan dari *Customer Care*.
- *Central Manager*, adalah *employee* yang membawahi semua kantor cabang perusahaan.

- Data tugas (*task*), merupakan jadwal harian untuk agen dan *customer care*.
- Data event dan iklan, merupakan acara-acara dan iklan-iklan yang diadakan atau dibuat perusahaan untuk tujuan promosi.

Berhubungan dengan data-data yang ditampilkan di atas, maka tujuan dari pengembangan perangkat lunak *Customer Relationship Management* ini adalah menangani pengelolaan data-data sebagai berikut:

1. Menangani **pengelolaan user**, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta memberikan hak akses kepada masing-masing *user* yang menggunakan perangkat lunak.
2. Menangani **pengelolaan produk asuransi**, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta hubungannya dengan *customer*.
3. Menangani **pengelolaan *customer***, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta mengelola hubungannya dengan *customer care* dan agen(sales).
4. Menangani **pengelolaan *employee***, digunakan untuk melakukan fungsi tambah, edit, hapus, dan cari terhadap data *employee*/pegawai.

Employee sendiri akan dibagi menjadi 3 jenis, yaitu Agen, *Customer Care* dan *Central Manager*.

5. Menangani **pengelolaan tugas/*task***. Digunakan untuk melakukan fungsi tambah, edit, hapus, serta manajemen terhadap daftar tugas yang diberikan dari pihak *customer care* terhadap agen.
6. Menangani **pengelolaan *mail***. Digunakan untuk melakukan fungsi tambah, edit, hapus, dan proses pengirimannya dari satu *user* ke *user* yang lain.
7. Menangani **pengelolaan *report***, digunakan untuk memperoleh informasi yang berhubungan dengan data-data dan pengelolaan yang telah dipapar sebelumnya.

1.3. Definisi Akronim dan Singkatan

Daftar definisi akronim dan singkatan:

Keyword/Phrase	Definisi
Agen	Pihak perusahaan yang bertugas untuk mencari dan menjaga *customer*.
Branch	Kantor cabang perusahaan.
Competitor	Perusahaan pesaing yang bergerak di bidang yang sama (asuransi).
CRM(Customer Relationsip Management)	Perangkat lunak yang digunakan untuk membantu dalam pengelolaan/manajemen terhadap relasi dengan *customer*.

Customer	Pengguna atau anggota yang terdaftar di dalam salah satu jenis fasilitas asuransi yang ditawarkan.
Customer Care	Pihak perusahaan yang bertugas untuk menjaga hubungan dengan *customer*.
DataBase	Merupakan kelompok data (arsip) yang saling berhubungan dan diorganisir sedemikianrupa agar dapat menghasilkan informasi dan dapat dimanfaatkan kembali dengan cepat dan mudah.
ERD	*Entity Relationship Diagram* merupakan teknis grafis/diagram yang menggambarkan *objek* dan hubungan antar *objek*.
Event .	Acara-acara yang diadakan oleh perusahaan untuk tujuan promosi.
Hak Akses	Hak yang dimiliki seorang *user* untuk menggunakan sistem yang diatur oleh *administrator* (dalam kasus ini bagian *Supervisor*).
Internet	Internet merupakan istilah umum yang dipakai untuk menunjuk *Network* global yang terdiri dari komputer dan layanan servis dengan sekitar 30 sampai 50 juta pemakai komputer dan puluhan layanan informasi termasuk e-mail, FTP, dan World Wide Web.

Login	Sebuah ketentuan (berupa *interface*) yang akan menyeleksi hak akses dari seorang *user* (kesesuaian antara *user name* dengan *password-nya*).
ORDBMS	*Object Relational DataBase Management System* atau sistem pengelolaan untuk *database* dengan konsep relasi antar *object*.
Premi	Uang iuran yang dibayarkan kepada pihak perusahaan asuransi untuk memenuhi kewajiban *customer*.
Prospect Customer	*Customer* yang sedang diusahakan oleh agen untuk *closing* (menjadi pengguna atau anggota produk asuransi yang ditawarkan).
Report	Laporan yang akan disajikan kepada seseorang atau pihak sesuai dengan kebutuhannya (berdasar hak aksesnya).
Server	Komputer yang menyediakan sumber daya bagi klien yang terhubung melalui jaringan.
SKPL	Merupakan spesifikasi kebutuhan dari perangkat lunak yang akan dikembangkan.
UML (Unified Modeling Language)	Merupakan bahasa standart untuk melakukan visualisasi, spesifikasi, pembangunan, dan dokumentasi untuk sebuah software.

User Interface	Merupakan antarmuka yang akan menghubungkan *user* dengan sebuah sistem/perangkat lunak yang ada.
Use Case Diagram	Merupakan diagram yang digunakan untuk membantu menemukan obyek, kelas, dan relasi, serta menggambarkan kebutuhan dengan melihat bagaimana sistem dibutuhkan dan siapa penggunanya.

1.4. Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

1. Alan M.Davis, Software Requirement, Prentice Hall, International Edition, 1993.
2. Bennet Simon, McRobb Steve, Farmer Ray, *Object-Oriented System Analysis and Design Using UML*, McGraw-Hill Companies, 2002.
3. Boggs Wendy, Boggs Michael, *Mastering UML with Rational Rose 2002*, SYBEX Inc, 2002.
4. Deitel, *C# How to Program*, Prentice-Hall Inc, 2002.
5. MSDN Library-October 2005, Microsoft, 2005.
6. Presman Roger S., *Rekayasa Perangkat Lunak*, McGraw-Hill Book Co., Andi Yogyakarta, 1997.

1.5. Deskripsi umum (*Overview*)

Secara garis besar, dokumen SKPL ini terdiri atas 3 bagian utama, pertama adalah bagian pendahuluan yang meliputi tujuan, lingkup masalah, definisi akronim dan singkatan yang digunakan, referensi, dan deskripsi umum tentang dokumen SKPL ini. Kemudian yang kedua adalah deskripsi umum/menyeluruh tentang produk yang menyangkut perspektif produk, fungsi produk, karakteristik, batasan-batasan, serta asumsi dan ketergantungan sistem tersebut. Untuk perspektif produk masih dibagi lagi menjadi beberapa bagian yaitu antarmuka pemakai, antarmuka keras, antarmuka perangkat lunak, antarmuka komunikasi, batasan memori, dan operasi. Yang ketiga menyangkut tentang kebutuhan-kebutuhan khusus beserta penjelasan lengkap dari sistem tersebut. Kebutuhan khusus yang dimaksud adalah kebutuhan antarmuka eksternal, kebutuhan fungsionalitas yang menjelaskan mengenai *use case* dari proses yang ada, untuk menentukan obyek, kelas, dan relasi untuk menentukan kebutuhan sistem dan penggunanya.

A.5 Fungsi *Mail Management* (***SKPL-CRM-A-05***) merupakan fungsi yang digunakan untuk melakukan pengelolaan mail, mencakup:

A.5.1 Fungsi *Create Mail* (***SKPL-CRM-A-05-01***) adalah fungsi yang digunakan untuk membuat dan mengirimkan *mail* ke *user* lain.

A.5.2 Fungsi *Delete Mail* (***SKPL-CRM-A-05-02***) adalah fungsi yang digunakan untuk menghapus *mail* yang dikirimkan kepada *user* yang bersangkutan.

A.6 Fungsi *Password Management* (***SKPL-CRM-A-06***) merupakan fungsi yang digunakan untuk mengubah *password* dari *user* yang bersangkutan.

A.7 Fungsi *Central Manager Management* (***SKPL-CRM-A-07***) adalah fungsi yang digunakan untuk mengelola data central manager dari perangkat lunak *CRM*, mencakup:

A.7.1 Fungsi *Add Central Manager* (***SKPL-CRM-A-02-01***) adalah fungsi yang digunakan untuk menambah central manager yang baru.

A.7.2 Fungsi *Edit Central Manager* (***SKPL-CRM-A-07-02***) adalah fungsi yang digunakan untuk memperbaiki kesalahan ataupun mengubah data central manager.

A.7.3 Fungsi *Delete Central Manager* (**SKPL-*CRM*-A-07-03**) adalah fungsi yang digunakan untuk menghapus data central manager yang sudah tidak memiliki hak akses.

A.7.4 Fungsi *Display Central Manager* (**SKPL-*CRM*-A-07-04**) adalah fungsi yang digunakan untuk melakukan pencarian terhadap central manager tertentu berdasarkan kategori tertentu.

B. *Customer Care*

B.1 Fungsi *Login* (**SKPL-*CRM*-A-01**) merupakan fungsi yang digunakan oleh bagian *customer care* untuk dapat masuk ke dalam perangkat lunak ini sesuai dengan haknya.

B.2 Fungsi *Customer Management* (**SKPL-*CRM*-B-02**) merupakan fungsi yang digunakan untuk mengelola data *customer* dari produk-produk asuransi yang ditawarkan, meliputi:

B.2.1 Fungsi *Add Customer* (**SKPL-*CRM*-B-02-01**) adalah fungsi yang digunakan untuk menambahkan data *customer* yang baru.

B.2.2 Fungsi *Edit Customer* (**SKPL-*CRM*-B-02-02**) adalah fungsi yang

digunakan untuk memperbaiki atau mengubah data *customer*.

B.2.3 Fungsi *Delete Customer* (**SKPL-*CRM*-B-02-03**) adalah fungsi yang digunakan untuk menghapus data *customer* yang sudah tidak lagi menjadi anggota salah satu produk asuransi.

B.2.4 Fungsi *Search Customer* (**SKPL-*CRM*-B-02-04**) adalah fungsi yang digunakan untuk pencarian terhadap data *customer* yang diperlukan.

B.3 Fungsi *Task Management* (**SKPL-*CRM*-B-03**) merupakan fungsi yang digunakan untuk melakukan pengelolaan terhadap tugas/*task*, baik itu harian, mingguan, maupun bulanan bagi agen maupun dirinya sendiri, meliputi:

B.3.1 Fungsi *Add task* (**SKPL-*CRM*-B-03-01**) adalah fungsi yang digunakan untuk memberikan tugas/*task* baru bagi agen maupun dirinya sendiri.

B.3.2 Fungsi *Edit task* (**SKPL-*CRM*-B-03-02**) adalah fungsi yang digunakan untuk merevisi tugas/*task* yang dia berikan kepada agen maupun tugas/*task* bagi dirinya sendiri.

B.3.3 Fungsi *Delete task* (**SKPL-*CRM*-B-03-03**) adalah fungsi yang digunakan untuk menghapus tugas/*task* yang dia berikan kepada agen maupun dirinya sendiri.

B.3.4 Fungsi *Show task* (**SKPL-*CRM*-B-03-04**) adalah fungsi untuk menampilkan data mengenai tugas/*task* yang diberikan kepada agen maupun dirinya sendiri.

B.4 Fungsi *Reporting* (**SKPL-*CRM*-B-04**) merupakan fungsi yang digunakan untuk mendapatkan informasi dari data-data yang diperlukan sesuai dengan keperluan dan haknya sebagai *customer care*. *Report* yang akan disajikan dalam bentuk tabel dan grafik ini, meliputi:

B.4.1 Fungsi *Customer Report* (**SKPL-*CRM*-B-04-01**) adalah fungsi yang digunakan untuk memperoleh informasi mengenai hal-hal yang berhubungan dengan *customer*, seperti data pribadi ataupun data yang berkaitan dengan agen-nya.

B.4.2 Fungsi *Agen Report* (**SKPL-*CRM*-B-04-02**) adalah fungsi yang digunakan untuk memperoleh informasi mengenai hal-hal yang berhubungan dengan agen, seperti

data pribadi ataupun data yang berkaitan dengan *customer* yang agen tersebut dapatkan.

B.5 Fungsi *Agen Management* (**SKPL-*CRM*-B-05**) merupakan fungsi yang digunakan untuk melakukan pengelolaan data agen perusahaan, meliputi:

B.5.1 Fungsi *Add Agen* (**SKPL-*CRM*-B-05-01**) adalah fungsi yang digunakan untuk menambah agen.

B.5.2 Fungsi *Edit Agen* (**SKPL-*CRM*-B-05-02**) adalah fungsi yang digunakan untuk mengedit data agen.

B.5.3 Fungsi *Delete Agen* (**SKPL-*CRM*-B-05-03**) adalah fungsi yang digunakan untuk menghapus data agen.

B.5.4 Fungsi *Search Agen* (**SKPL-*CRM*-B-05-04**) adalah fungsi yang digunakan untuk mencari dan menampilkan data agen.

B.6 Fungsi *Mail Management* (***SKPL-CRM*-A-05**) merupakan fungsi yang digunakan untuk melakukan pengelolaan mail, mencakup:

B.6.1 Fungsi *Create Mail* (***SKPL-CRM*-A-05-01**) adalah fungsi yang digunakan untuk membuat dan mengirimkan *mail* ke *user* lain.

B.6.2 Fungsi *Delete Mail* (**SKPL-CRM-A-05-02**) adalah fungsi yang digunakan untuk menghapus *mail* yang dikirimkan kepada *user* yang bersangkutan.

B.7 Fungsi *Password Management* (**SKPL-CRM-A-06**) merupakan fungsi yang digunakan untuk mengubah *password* dari *user* yang bersangkutan.

C. Agen

C.1 Fungsi *Login* (**SKPL-CRM-A-01**) merupakan fungsi yang digunakan oleh agen untuk dapat masuk ke dalam perangkat lunak ini sesuai dengan haknya.

C.2 Fungsi *Task Management* (**SKPL-CRM-B-02**) merupakan fungsi yang digunakan untuk melakukan pengelolaan terhadap tugas/*task*, baik itu harian, mingguan, maupun bulanan bagi dirinya sendiri, meliputi:

C.2.1 Fungsi *Add Task* (**SKPL-CRM-B-02-01**) adalah fungsi yang digunakan untuk memberikan tugas/ *task* baru yang harus dilakukan.

C.2.2 Fungsi *Edit Task* (**SKPL-CRM-B-02-02**) adalah fungsi yang digunakan untuk merevisi tugas/ *task* yang harus dilakukan.

C.2.3 Fungsi *Delete Task* (**SKPL-*CRM*-B-02-03**) adalah fungsi yang digunakan untuk menghapus tugas/ *task* yang telah dilakukan.

C.2.4 Fungsi *Show Task* (**SKPL-*CRM*-B-02-04**) adalah fungsi untuk menampilkan data mengenai tugas/ *task* yang harus dia lakukan.

C.3 Fungsi *Reporting* (**SKPL-*CRM*-B-03**) merupakan fungsi yang digunakan untuk mendapatkan informasi dari data-data yang diperlukan sesuai dengan keperluan dan haknya sebagai agen. *Report* yang akan disajikan dalam bentuk tabel dan grafik ini, meliputi:

C.3.1 Fungsi *Customer Report* (**SKPL-*CRM*-B-03-01**) adalah fungsi yang digunakan untuk memperoleh informasi mengenai hal-hal yang berhubungan dengan *customer* yang dia peroleh, misalnya data pribadi.

C.3.2 Fungsi *Agen Report* (**SKPL-*CRM*-B-03-03**) adalah fungsi yang digunakan untuk memperoleh informasi mengenai hal-hal yang berhubungan dengan agen, seperti data pribadi ataupun data yang berkaitan dengan *customer* yang agen tersebut dapatkan.

C.4 Fungsi *Mail Management* (**SKPL-CRM-A-05**) merupakan fungsi yang digunakan untuk melakukan pengelolaan mail, mencakup:

C.4.1 Fungsi *Create Mail* (**SKPL-CRM-A-05-01**) adalah fungsi yang digunakan untuk membuat dan mengirimkan *mail* ke *user* lain.

C.4.2 Fungsi *Delete Mail* (**SKPL-CRM-A-05-02**) adalah fungsi yang digunakan untuk menghapus *mail* yang dikirimkan kepada *user* yang bersangkutan.

C.5 Fungsi *Password Management* (**SKPL-CRM-A-06**) merupakan fungsi yang digunakan untuk mengubah *password* dari *user* yang bersangkutan.

D. *Customer*

D.1 Fungsi *Login* (**SKPL-CRM-A-01**) merupakan fungsi yang digunakan oleh agen untuk dapat masuk ke dalam perangkat lunak ini sesuai dengan haknya.

D.2 Fungsi *Registration* (**SKPL-CRM-D-02**) adalah fungsi yang digunakan oleh *customer* untuk mendaftarkan diri sebagai pengguna perangkat lunak (memiliki *account*).

D.3 Fungsi *Account Report* (**SKPL-CRM-D-03**) merupakan fungsi yang digunakan oleh *customer* yang telah melakukan registrasi

sebelumnya (memiliki *account*) untuk mendapatkan informasi mengenai statusnya sebagai *customer*.

D.4 Fungsi *Mail Management* (**SKPL-CRM-A-05**) merupakan fungsi yang digunakan untuk melakukan pengelolaan mail, mencakup:

D.4.1 Fungsi *Create Mail* (**SKPL-CRM-A-05-01**) adalah fungsi yang digunakan untuk membuat dan mengirimkan *mail* ke *user* lain.

D.4.2 Fungsi *Delete Mail* (**SKPL-CRM-A-05-02**) adalah fungsi yang digunakan untuk menghapus *mail* yang dikirimkan kepada *user* yang bersangkutan.

D.5 Fungsi *Password Management* (**SKPL-CRM-A-06**) merupakan fungsi yang digunakan untuk mengubah *password* dari *user* yang bersangkutan.

E. *Central Manager*

E.1 Fungsi *Login* (**SKPL-CRM-A-01**) merupakan fungsi yang digunakan oleh agen untuk dapat masuk ke dalam perangkat lunak ini sesuai dengan haknya.

E.2 Fungsi *Reporting* (**SKPL-CRM-B-02**) merupakan fungsi yang digunakan untuk mendapatkan informasi dari data-data yang diperlukan sesuai dengan keperluan

dan haknya sebagai *Central Manager*. *Report* ini meliputi:

E.2.1 Fungsi *Product Report* (**SKPL-*CRM*-E-02-01**) adalah fungsi untuk mendapatkan informasi mengenai perbandingan antara produk yang paling diminati ataupun juga sebaliknya.

E.2.2 Fungsi *All Branch Report* (**SKPL-*CRM*-E-02-02**) adalah fungsi yang digunakan untuk mengetahui posisi semua kantor cabang, baik dalam hal keuangan maupun *customer*.

E.3 Fungsi *Mail Management* (**SKPL-*CRM*-A-05**) merupakan fungsi yang digunakan untuk melakukan pengelolaan mail, mencakup:

E.3.1 Fungsi *Create Mail* (**SKPL-*CRM*-A-05-01**) adalah fungsi yang digunakan untuk membuat dan mengirimkan *mail* ke *user* lain.

E.3.2 Fungsi *Delete Mail* (**SKPL-*CRM*-A-05-02**) adalah fungsi yang digunakan untuk menghapus *mail* yang dikirimkan kepada *user* yang bersangkutan.

E.4 Fungsi *Password Management* (**SKPL-*CRM*-A-06**) merupakan fungsi yang digunakan untuk mengubah *password* dari *user* yang bersangkutan.

2.4. Karakteristik Pengguna

Pengguna (*user*) perangkat lunak *CRM* ini adalah *Administrator* dan *Customer Care* dari PT. Asuransi Allianz Life Indonesia cabang Yogyakarta yang memiliki karakteristik sebagai berikut:

- *Administrator*
  1. Memahami pengoperasian komputer secara aktif.
  2. Memahami sistem operasi komputer tempat perangkat lunak dijalankan.
  3. Mengerti dan memahami konsep *CRM* dan perangkat lunak yang digunakan.
  4. Memahami pengelolaan *user*, *employee*, dan *branch* pada perangkat lunak *CRM* ini.
  5. Memahami *bussiness process* dari perusahaan yang bersangkutan.

- *Customer Care*
  1. Memahami pengoperasian computer secara aktif.
  2. Memahami sistem operasi komputer tempat perangkat lunak dijalankan.
  3. Mengerti dan memahami konsep *CRM* dan perangkat lunak yang digunakan.
  4. Memahami pengoperasian komputer serta perangkat tambahan komputer yang digunakan.
  5. Memahami *bussiness process* dari perusahaan yang bersangkutan.

- Agen
  1. Memahami pengoperasian computer secara aktif.
  2. Mengerti dan memahami konsep *CRM* dan perangkat lunak yang digunakan.
  3. Memahami pengoperasian komputer serta perangkat tambahan komputer yang digunakan.
  4. Memahami *bussiness process* dari perusahaan yang bersangkutan.
- *Customer*
  1. Memahami pengoperasian computer secara aktif.
  2. Memahami pengoperasian komputer serta perangkat tambahan komputer yang digunakan.
- *Central Manager*
  1. Memahami pengoperasian computer secara aktif.
  2. Memahami sistem operasi komputer tempat perangkat lunak dijalankan.
  3. Mengerti dan memahami konsep *CRM* dan perangkat lunak yang digunakan.
  4. Memahami pengoperasian komputer serta perangkat tambahan komputer yang digunakan.
  5. Memahami *bussiness process* dari perusahaan yang bersangkutan.

2.5. Batasan-batasan

Batasan-batasan dalam pengembangan perangkat lunak *CRM* ini adalah:

1. Kebijaksanaan Umum

   Berpedoman pada tujuan dari pengembangan perangkat lunak *CRM*.

2. Keterbatasan perangkat keras

   Ditentukan kemudian setelah pengembang mengetahui ketersediaan perangkat keras pada pelanggan.

3. Kebutuhan keandalan

   Pengembangan perangkat lunak ini dibatasi pada kemudahan penggunaan dan kecepatan dalam proses pengolahannya.

2.6. Asumsi dan Ketergantungan

Asumsi yang digunakan dalam perangkat lunak *CRM* ini yaitu:

- Tersedia perangkat lunak sesuai kebutuhan untuk pengoperasian produk perangkat lunak *CRM*.
- Tersedia komputer *server* dengan spesifikasi *prosesor* 3.0 *GHz*, memori primer minimal 512 *MB*, spasi yang tersimpan dalam media penyimpanan sekunder *server* adalah 256 *MB*, *Internet Modem* sebagai perangkat komunikasi.
- Tersedia pengendali *ODBC* atau *ODBC driver* untuk antarmuka *ORDBMS Oracle 9i*.

- Sudah ada sistem kerja *Administrator* pada *server*.
- *Entry* data informasi oleh *Administrator* sifatnya adalah *valid*.

3. Kebutuhan khusus

3.1 Kebutuhan Antarmuka Eksternal

Kebutuhan antar muka eksternal pada perangkat lunak *CRM* ini meliputi kebutuhan antarmuka pemakai, antarmuka perangkat keras, antarmuka perangkat lunak, dan antarmuka komunikasi.

3.1.1 Antarmuka Pemakai

Jenis antarmuka yang digunakan oleh seluruh *user* (administrator, *customer care*, agen, *customer*, dan *central manager*) pada perangkat lunak *CRM* ini adalah jenis antarmuka *web based*. Bagi administrator, antarmuka yang disediakan adalah antarmuka yang digunakan untuk melakukan *login*, *customer care management*, *central manager*, *product management*, *mail management*, dan *branch management*. Bagi *Customer care*, antarmuka yang disediakan digunakan untuk melakukan login, *customer management*, *task management*, *mail management* dan mendapatkan *report* yang diperlukan.

Bagi agen, antarmuka yang disediakan adalah antarmuka yang digunakan untuk melakukan login, *task management*, *mail*

management dan mendapatkan *report* yang diperlukan. Bagi *customer*, disediakan antarmuka untuk melakukan login, registrasi untuk mendapatkan *account*, *mail management*, dan memperoleh *report* mengenai posisinya dalam asuransi yang bersangkutan. Sedangkan untuk *central manager*, disediakan antarmuka untuk melakukan login, *mail management* dan mendapatkan *report* mengenai posisi keuangan dan *customer* dari seluruh kantor cabang.

3.1.2 Antarmuka Perangkat Keras

Piranti antarmuka perangkat keras yang dibutuhkan dalam perangkat lunak *CRM* ini adalah:

1. *Personal Computer IBM Compatible P-4*
2. RAM minimal 256 MB.
3. *Harddisk.*
4. *Keyboard, mouse, monitor* dan *printer.*
5. *Internet Modem.*

3.1.3 Antarmuka Perangkat Lunak

Perangkat lunak yang diperlukan dalam pengoperasian perangkat lunak *CRM* ini adalah sebagai berikut:

1. Nama : Windows 2000 Server
   Sumber : Microsoft
   Sebagai sistem operasi dimana perangkat lunak *CRM* dijalankan.

2. Nama : Oracle 9i
   Sumber : Oracle
   Sebagai wadah *ORDBMS* yang dibutuhkan untuk mengoperasikan perangkat lunak *CRM* ini.
3. Nama : ASP .NET 2005
   (bahasa pemrograman C#)
   Sumber : Microsoft
   Sebagai *tool* pembuatan program dan *user interface*.
4. Nama : Internet Information Server (IIS)
   Sumber : Microsoft
   Sebagai *Web Server*.
5. Nama : Internet explorer 5.5
   Sumber : Microsoft
   Sebagai *internet browser*.

3.1.4 Antarmuka Komunikasi

Antarmuka komunikasi perangkat lunak CRM ini menggunakan protocol HTTP.

3.1.5 Batasan Memori

Batasan memori primer yang diperlukan dalam operasional *CRM* ini adalah khusus untuk *server RAM* minimal 512*MB*, direkomendasikan 1*GB*. Kemudian untuk memori *Harddisk* direkomendasikan minimal 80*GB* tersedia. Sedangkan untuk *browser*, *RAM* minimal 128*MB*,

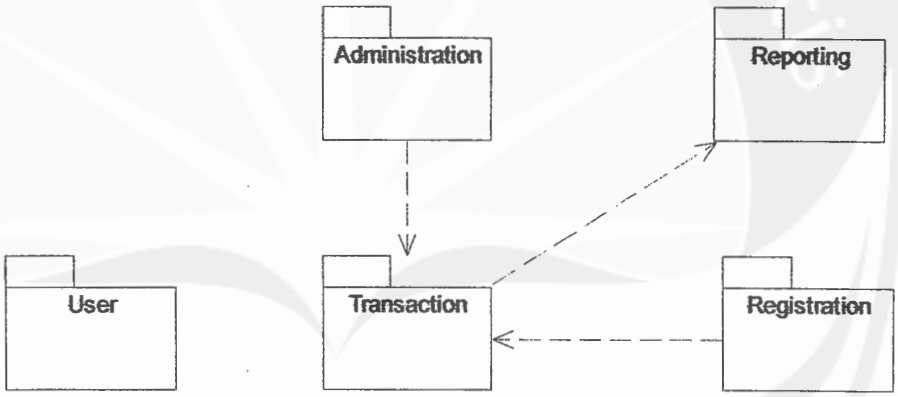
direkomendasikan 256*MB*. Dan untuk memori Harddisk direkomendasikan minimal 20*GB*.

3.1.6 Operasi

Variasi model operasi yang disediakan dalam perangkat lunak *CRM* ini adalah mode Interaktif Terbatas, yaitu *user* hanya dapat berinteraksi (memperoleh informasi dan melakukan perubahan) dengan perangkat lunak sesuai dengan posisi dan *role*-nya saja. Untuk akses selain wewenangnya, tidak diperkenankan.

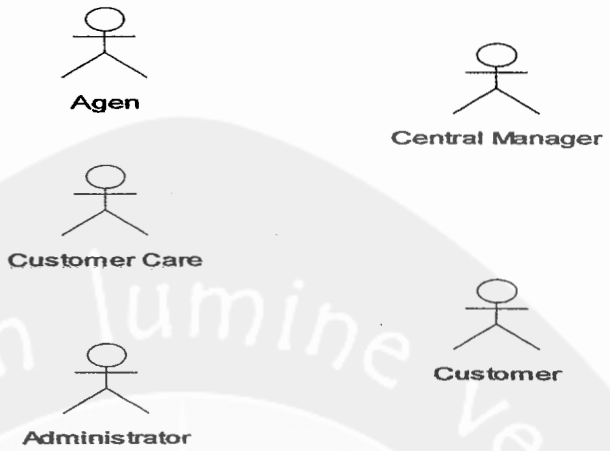
3.2 Kebutuhan fungsionalitas Perangkat Lunak

3.2.1 *Package Dependencies*



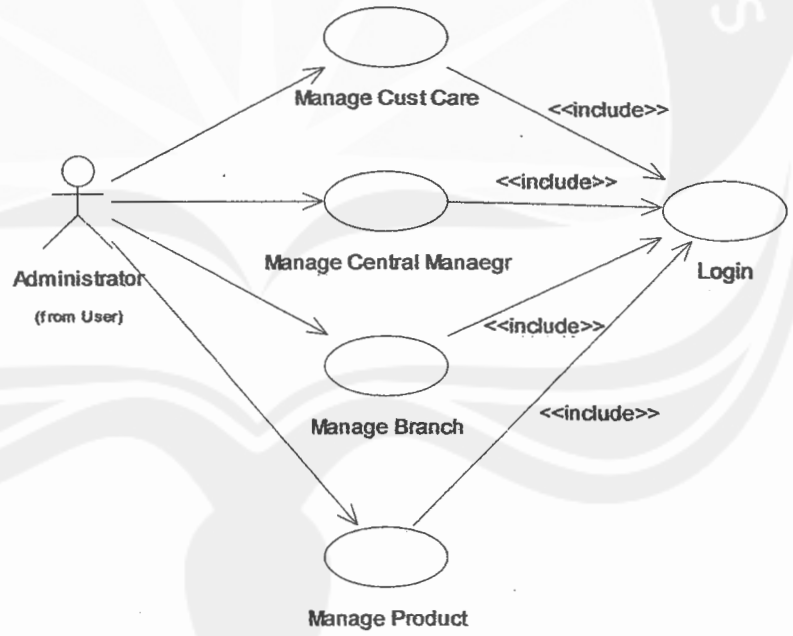
Gambar 2. *Package Dependencies*

3.2.2 *Use Case for User Package*



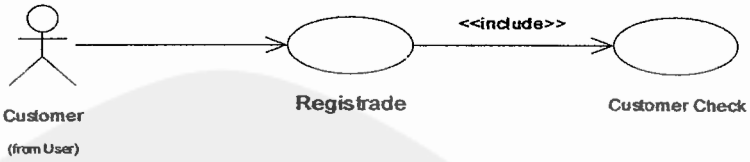
Gambar 3. *Use Case for User Package*

3.2.3 *Use Case for Administration Package*



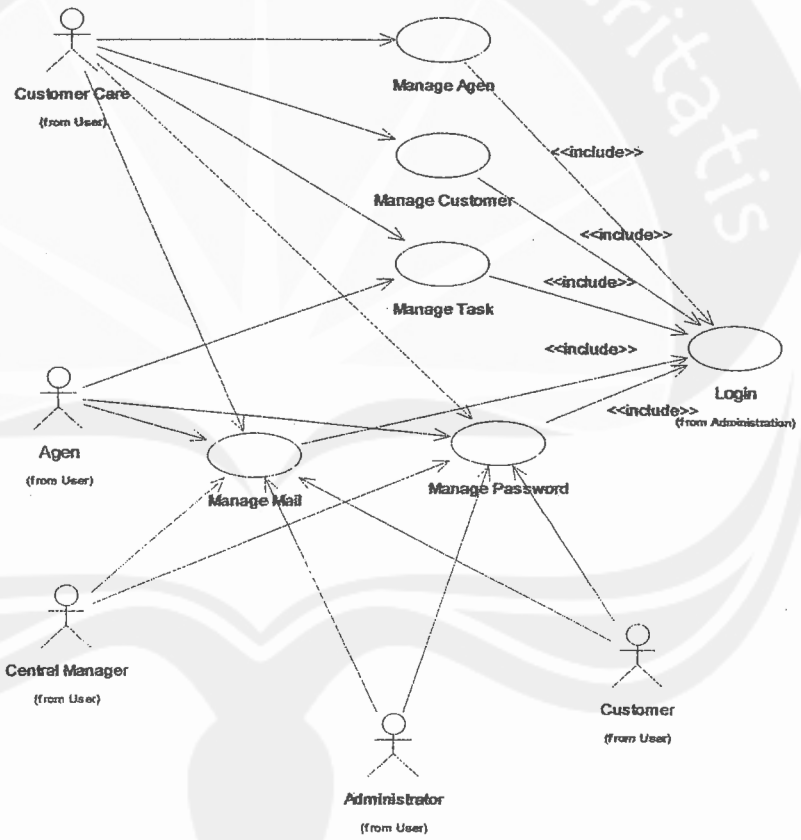
Gambar 4. *Use Case for Administration Package*

3.2.4 *Use Case for Registration Package*



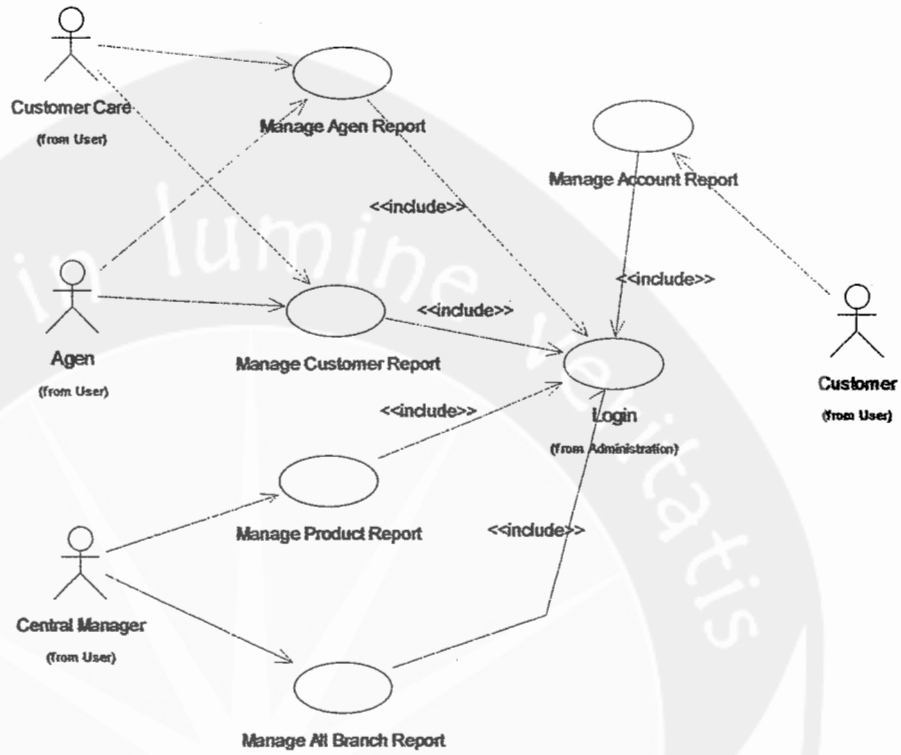
Gambar 5. *Use Case for Registration Package*

3.2.5 *Use Case for Transaction Package*



Gambar 6. *Use Case for Transaction Package*

3.2.6 *Use Case for Reporting Package*



Gambar 7. *Use Case for Reporting Package*

4. Spesifikasi Rinci Kebutuhan

4.1. Spesifikasi Kebutuhan Fungsionalitas

4.1.1. *Use Case Spesification* : *Login*

1. ***Brief Description***

   Use case ini digunakan oleh aktor untuk memperoleh akses ke perangkat lunak. Login didasarkan pada sebuah *user name* yang sifatnya unik dan *password* yang berupa serangkaian karakter.

2. ***Primary Actor***

   1. Administrator
   2. Agen
   3. *Customer*
   4. *Customer Care*
   5. *Central Manager*

3. ***Supporting Actor***

   none

4. ***Basic Flow***

   1. *Use Case* ini dimulai ketika aktor memilih untuk melakukan *login*.
   2. Sistem menampilkan antarmuka untuk *login*.
   3. Aktor memasukkan *user name* dan *password*.
   4. Sistem memeriksa *user name* dan *password* yang diinputkan aktor.

      E-1 *Password* atau *user name* tidak sesuai.
   5. Sistem memberikan akses ke aktor.
   6. *Use case* ini selesai.

5. ***Alternative Flow***

none

6. ***Error Flow***

E-1 *Password* atau *user name* tidak sesuai

1. Sistem menampilkan peringatan bahwa *user name* atau *password* tidak sesuai.
2. Kembali ke *basic flow* langkah ke-3.

7. ***PreConditions***

none

8. ***PostConditions***

1. Aktor memasuki sistem dan dapat menggunakan fungsi-fungsi yang tersedia sesuai dengan haknya.

4.1.2. *Use Case Spesification* : *Manage Cust Care*

1. ***Brief Description***

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data pengguna sistem beserta hak aksesnya masing-masing. Aktor dapat melakukan *add*, edit, *delete*, dan *search* data *customer care*.

2. ***Primary Actor***

1. Administrator

3. ***Supporting Actor***

none

4. ***Basic Flow***

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data *customer care*.

2. Sistem memberikan pilihan untuk melakukan *add*, edit, *delete*, atau *search* data *customer care*.
3. Aktor memilih untuk melakukan *add* data *customer care*.
   A-1 Aktor memilih untuk melakukan edit data *customer care*.
   A-2 Aktor memilih untuk melakukan *delete* data *customer care*.
   A-3 Aktor memilih untuk melakukan *search* data *customer care*.
4. Aktor memasukkan data *customer care* yang baru.
5. Sistem melakukan pengecekan terhadap data *customer care* yang baru.
   E-1 Data *customer care* baru yang diinputkan salah.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. ***Alternatif Flow***

A-1 Aktor memilih untuk melakukan edit data *customer care*.
1. Sistem menampilkan data *customer care* yang akan diedit.
2. Aktor mengedit data *customer care* yang telah ditampilkan.
3. Aktor meminta sistem untuk menyimpan data *customer care* yang telah diedit.

4. Sistem melakukan pengecekan terhadap data *customer care* yang telah diedit.

E-2 Data *customer care* yang telah diedit salah.

5. Sistem menyimpan data *customer care* yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

A-2 Aktor memilih untuk melakukan *delete* data *customer care*.

1. Sistem menampilkan data *customer care* yang akan dihapus.
2. Aktor melakukan konfirmasi mengenai data *customer care* yang akan dihapus.
3. Sistem menghapus data *customer care* yang dimaksud.
4. Berlanjut ke *basic flow* langkah ke-7.

A-3 Aktor memilih untuk melakukan *search* data *customer care*.

1. Aktor memasukkan data *customer care* yang ingin dicari sesuai dengan kriterianya.
2. Sistem melakukan pengecekan terhadap *database customer care*.

E-3 Data *customer care* tidak ditemukan dalam *database*.

3. Sistem menampilkan data *customer care* sesuai dengan yang dicari.
4. *Use case* selesai.

6. *Error Flow*

E-1 Data *customer care* baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data *customer care* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data *customer care* yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data *customer care* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

E-3 Data *customer care* tidak ditemukan dalam *database*.

1. Sistem menampilkan informasi bahwa data *customer care* yang dicari tidak ada.
2. Kembali ke *alternatif flow* A-3 langkah ke-1.

7. PreConditions

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. PostConditions

1. Data *customer care* di *database* telah ter-*update*.

4.1.3. *Use Case Spesification* : *Manage Central Manager*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data pengguna sistem beserta hak aksesnya masing-masing. Aktor dapat melakukan *add*, edit, *delete*, dan *display* data *central manager*.

2. *Primary Actor*

1. Administrator

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data *central manager*.
2. Sistem memberikan pilihan untuk melakukan *add*, edit, *delete*, atau *display* data *central manager*.
3. Aktor memilih untuk melakukan *add* data *central manager*.
   - A-1 Aktor memilih untuk melakukan edit data *central manager*.
   - A-2 Aktor memilih untuk melakukan *delete* data *central manager*.
   - A-3 Aktor memilih untuk melakukan *display* data *central manager*.
4. Aktor memasukkan data *central manager* yang baru.

5. Sistem melakukan pengecekan terhadap data *central manager* yang baru.

E-1 Data *central manager* baru yang diinputkan salah.

6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. Alternatif Flow

A-1 Aktor memilih untuk melakukan edit data *central manager*.

1. Sistem menampilkan data *central manager* yang akan diedit.
2. Aktor mengedit data *central manager* yang telah ditampilkan.
3. Aktor meminta sistem untuk menyimpan data *central manager* yang telah diedit.
4. Sistem melakukan pengecekan terhadap data *central manager* yang telah diedit.

E-2 Data *central manager* yang telah diedit salah.

5. Sistem menyimpan data *central manager* yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

A-2 Aktor memilih untuk melakukan *delete* data *central manager*.

1. Sistem menampilkan data *central manager* yang akan dihapus.

2. Aktor melakukan konfirmasi mengenai data *central manager* yang akan dihapus.
3. Sistem menghapus data *central manager* yang dimaksud.
4. Berlanjut ke *basic flow* langkah ke-7.

A-3 Aktor memilih untuk melakukan *display* data *central manager*.
1. Sistem menampilkan data *central manager* yang pada saat itu menjabat sebagai *central manager*.
2. *Use case* selesai.

6. Error Flow

E-1 Data *central manager* baru yang diinputkan salah.
1. Sistem menampilkan peringatan bahwa data *central manager* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data *central manager* yang telah diedit salah.
1. Sistem menampilkan peringatan bahwa data *central manager* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

7. PreConditions

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. *PostConditions*

1. Data *central manager* di *database* telah ter-*update*.

4.1.4. *Use Case Spesification* : *Manage Branch*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data *Branch* atau kantor cabang perusahaan. Aktor dapat melakukan *add*, edit, dan *delete* data *branch*.

2. *Primary Actor*

1. Administrator

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data *branch*.
2. Sistem memberikan pilihan untuk melakukan *add*, edit, atau *delete* data *branch*.
3. Aktor memilih untuk melakukan *add* data *branch*.
   - A-1 Aktor memilih untuk melakukan edit data *branch*.
   - A-2 Aktor memilih untuk melakukan *delete* data *branch*.
4. Aktor memasukkan data *branch* yang baru.

5. Sistem melakukan pengecekan terhadap data *branch* yang baru.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. ***Alternatif Flow***

A-1 Aktor memilih untuk melakukan edit data *branch*.

1. Sistem menampilkan data *branch* yang akan diedit.
2. Aktor mengedit data *branch* yang telah ditampilkan.
3. Aktor meminta sistem untuk menyimpan data *branch* yang telah diedit.
4. Sistem melakukan pengecekan terhadap data *branch* yang telah diedit.

E-2 Data *branch* yang telah diedit salah.

5. Sistem menyimpan data *branch* yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

A-2 Aktor memilih untuk melakukan *delete* data *branch*.

1. Sistem menampilkan data *branch* yang akan dihapus.
2. Aktor melakukan konfirmasi mengenai data *branch* yang akan dihapus.
3. Sistem menghapus data *branch* yang dimaksud.

4. Berlanjut ke *basic flow* langkah ke-7.

6. *Error Flow*

E-1 Data *branch* baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data *branch* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data *branch* yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data *branch* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

7. *PreConditions*

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. *PostConditions*

1. Data *branch* di *database* telah ter-*update*.

4.1.5. *Use Case Spesification* : *Manage Product*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data *product* yang dikeluarkan perusahaan. Aktor dapat melakukan *add* dan edit data *product* asuransi.

2. *Primary Actor*

1. Administrator

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data *product*.
2. Sistem memberikan pilihan untuk melakukan *add* atau edit data *product*.
3. Aktor memilih untuk melakukan *add* data *branch*.

   A-1 Aktor memilih untuk melakukan edit data *product*.
4. Aktor memasukkan data *product* yang baru.
5. Sistem melakukan pengecekan terhadap data *product* yang baru.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. *Flow*

A-1 Aktor memilih untuk melakukan edit data *product*.

1. Sistem menampilkan data *product* yang akan diedit.
2. Aktor mengedit data *product* yang telah ditampilkan.
3. Aktor meminta sistem untuk menyimpan data *product* yang telah diedit.
4. Sistem melakukan pengecekan terhadap data *product* yang telah diedit.

   E-2 Data *product* yang telah diedit salah.

5. Sistem menyimpan data *product* yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

6. *Error Flow*

E-1 Data *product* baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data *product* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-2.

E-2 Data *product* yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data *product* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

7. *PreConditions*

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. *PostConditions*

1. Data *product* di *database* telah ter-*update*.

4.1.6. *Use Case Spesification* : *Registrade*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan registrasi atau pendaftaran agar mendapatkan sebuah *account* dalam sistem. *Account* ini akan digunakan oleh aktor untuk masuk ke dalam sistem.

2. *Primary Actor*

1. *Customer*

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use Case* ini dimulai ketika user memilih untuk melakukan registrasi.
2. Sistem akan menampilkan antarmuka untuk melakukan registrasi.
3. Aktor memasukkan data registrasi.
4. Sistem memeriksa data registrasi yang diinputkan aktor.

   E-1 Data yang dimasukkan tidak *valid.*
5. Sistem memberikan konfirmasi bahwa proses registrasi telah berhasil.
6. *Use case* selesai.

5. *Alternatif Flow*

none

6. *Error Flow*

E-1 Data yang dimasukkan tidak *valid.*

1. Sistem akan menampilkan peringatan bahwa masukan data untuk registrasi tidak *valid.*
2. Kembali ke *basic flow* langkah ke-3.

7. *PreConditions*

1. Aktor telah memasuki sistem.

8. ***PostConditions***

1. Aktor dapat memasuki sistem dan dapat menggunakan fungsi-fungsi yang tersedia sesuai dengan haknya.

4.1.7. ***Use Case Spesification*** : *Customer Check*

1. ***Brief Description***

Use case ini digunakan oleh sistem untuk melakukan pengecekan apakah aktor (*customer*) adalah benar merupakan anggota/*customer* dari perusahaan atau bukan. Apabila aktor benar adalah customer perusahaan, maka selanjutnya akan dapat melakukan proses registrasi.

2. ***Primary Actor***

1. *Customer*

3. ***Supporting Actor***

none

4. ***Basic Flow***

1. *Use Case* ini dimulai ketika aktor memilih untuk melakukan registrasi.
2. Sistem akan menampilkan antarmuka untuk melakukan registrasi.
3. Aktor memasukkan data registrasi.
4. Sistem memeriksa apakah aktor terdaftar sebagai customer atau tidak.

E-1 Aktor bukan sebagai *customer*.

5. Sistem akan melanjutkan ke proses registrasi.
6. *Use case* selesai.

5. *Alternatif Flow*

none

6. *Error Flow*

E-1 Aktor bukan sebagai *customer*.

1. Sistem akan menampilkan peringatan bahwa user bukanlah *customer* dari perusahaan.
2. Kembali ke *basic flow* langkah ke-3.

7. *PreConditions*

1. Aktor telah memasuki sistem.

8. *PostConditions*

1. Aktor diperbolehkan untuk melakukan proses registrasi untuk mendapatkan *account* yang digunakan untuk proses selanjutnya.

4.1.8. *Use Case Spesification* : *Manage Agen*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data pengguna sistem beserta hak aksesnya masing-masing. Aktor dapat melakukan *add*, edit, *delete*, dan *search* data agen.

2. *Primary Actor*

1. *Customer Care*

3. *Supporting Actor*

None

4. *Basic Flow*

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data agen.
2. Sistem memberikan pilihan untuk melakukan *add*, edit, *delete*, atau *search* data agen.
3. Aktor memilih untuk melakukan *add* data agen.
   - A-1 Aktor memilih untuk melakukan edit data agen.
   - A-2 Aktor memilih untuk melakukan *delete* data agen.
   - A-3 Aktor memilih untuk melakukan *search* data agen.
4. Aktor memasukkan data agen yang baru
5. Sistem melakukan pengecekan terhadap data agen yang baru.
   - E-1 Data agen baru yang diinputkan salah.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. *Alternatif Flow*

A-1 Aktor memilih untuk melakukan edit data agen.

1. Sistem menampilkan data agen yang akan diedit.
2. Aktor mengedit data agen yang telah ditampilkan.

3. Aktor meminta sistem untuk menyimpan data agen yang telah diedit.
4. Sistem melakukan pengecekan terhadap data agen yang telah diedit.

E-2 Data agen yang telah diedit salah.

5. Sistem menyimpan data agen yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

A-2 Aktor memilih untuk melakukan *delete* data *user*.

1. Sistem menampilkan data agen yang akan dihapus.
2. Aktor melakukan konfirmasi mengenai data agen yang akan dihapus.
3. Sistem menghapus data agen yang dimaksud.
4. Berlanjut ke *basic flow* langkah ke-7.

A-3 Aktor memilih untuk melakukan *search* data agen.

1. Aktor memasukkan data agen yang ingin dicari sesuai dengan kriterianya.
2. Sistem melakukan pengecekan terhadap *database* agen.

E-3 Data agen tidak ditemukan dalam *database*.

3. Sistem menampilkan data agen sesuai dengan yang dicari.
4. *Use case* selesai.

6. *Error Flow*

E-1 Data agen baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data agen baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data agen yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data agen yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

E-3 Data agen tidak ditemukan dalam *database*.

1. Sistem menampilkan informasi bahwa data agen yang dicari tidak ada.
2. Kembali ke *alternatif flow* A-3 langkah ke-1.

7. PreConditions

2. *Use Case* Login telah dilakukan.
3. Aktor telah memasuki sistem.

8. PostConditions

1. Data agen di *database* telah ter-*update*.

4.1.9. *Use Case Spesification* : *Manage Customer*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk melakukan pengelolaan terhadap data pengguna sistem beserta hak aksesnya masing-masing.

Aktor dapat melakukan *add*, edit, *delete*, dan *search* data *customer*.

2. *Primary Actor*

1. *Customer Care*

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use case* dimulai ketika aktor memilih untuk melakukan pengelolaan data *customer*.
2. Sistem memberikan pilihan untuk melakukan *add*, edit, *delete*, atau *search* data *customer*.
3. Aktor memilih untuk melakukan *add* data *customer*.
   - A-1 Aktor memilih untuk melakukan edit data *customer*.
   - A-2 Aktor memilih untuk melakukan *delete* data *customer*.
   - A-3 Aktor memilih untuk melakukan *search* data *customer*.
4. Aktor memasukkan data *customer* yang baru
5. Sistem melakukan pengecekan terhadap data *customer* yang baru.
   - E-1 Data *customer* baru yang diinputkan salah.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. *Alternatif Flow*

A-1 Aktor memilih untuk melakukan edit data *customer*.

1. Sistem menampilkan data *customer* yang akan diedit.
2. Aktor mengedit data *customer* yang telah ditampilkan.
3. Aktor meminta sistem untuk menyimpan data *customer* yang telah diedit.
4. Sistem melakukan pengecekan terhadap data *customer* yang telah diedit.

   E-2 Data *customer* yang telah diedit salah.
5. Sistem menyimpan data *customer* yang telah diedit.
6. Berlanjut ke *basic flow* langkah ke-7.

A=2 Aktor memilih untuk melakukan *delete* data *customer*.

1. Sistem menampilkan data *customer* yang akan dihapus.
2. Aktor melakukan konfirmasi mengenai data *customer* yang akan dihapus.
3. Sistem menghapus data *customer* yang dimaksud.
4. Berlanjut ke *basic flow* langkah ke=7.

A-3 Aktor memilih untuk melakukan *search* data *customer*.

1. Aktor memasukkan data *customer* yang ingin dicari sesuai dengan kriterianya.

2. Sistem melakukan pengecekan terhadap *database customer*.

   E-3 Data *customer* tidak ditemukan dalam *database*.

3. Sistem menampilkan data *customer* sesuai dengan yang dicari.
4. *Use case* selesai.

6. *Error Flow*

E-1 Data *customer* baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data *customer* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data *customer* yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data *customer* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

E-3 Data *customer* tidak ditemukan dalam *database*.

1. Sistem menampilkan informasi bahwa data *customer* yang dicari tidak ada.
2. Kembali ke *alternatif flow* A-3 langkah ke-1.

7. *PreConditions*

1. *Use Case* Login telah dilakukan.
2. Aktor telah memasuki sistem.

8. ***PostConditions***

1. Data *customer* di *database* telah ter-*update*.

4.1.10. ***Use Case Spesification*** : *Manage Task*

1. ***Brief Description***

Use case ini digunakan oleh aktor untuk melakukan pengelolaan tugas/*task* yang akan dilakukan oleh masing-masing aktor. Aktor dapat melakukan *add*, edit, *delete*, dan *display* data *task*.

2. ***Primary Actor***

1. *Customer care*
2. Agen

3. ***Supporting Actor***

none

4. ***Basic Flow***

1. *Use Case* ini dimulai ketika aktor memilih untuk melakukan pengelolaan *task*.
2. Sistem akan menampilkan antarmuka untuk melakukan pengelolaan *task*.
3. Aktor memilih untuk melakukan *add task*.

A-1 Aktor memilih untuk melakukan edit *task*.

A-2 Aktor memilih untuk melakukan *delete task*.

A-3 Aktor memilih untuk melakukan *display task*.

4. Aktor memasukkan data *task* yang baru.
5. Sistem memeriksa data *task* yang diinputkan aktor.

   E-1 Data yang dimasukkan tidak *valid*.
6. Sistem menyimpan data baru ke dalam *database*.
7. *Use case* selesai.

5. ***Alternatif Flow***

   A-1 Aktor memilih untuk melakukan edit data *task*.

   1. Sistem menampilkan data *task* yang akan diedit.
   2. Aktor mengedit data *task* yang telah ditampilkan.
   3. Aktor meminta sistem untuk menyimpan data *task* yang telah diedit.
   4. Sistem melakukan pengecekan terhadap data *task* yang telah diedit.

      E-2 Data *task* yang telah diedit salah.
   5. Sistem menyimpan data *task* yang telah diedit.
   6. Berlanjut ke *basic flow* langkah ke-7.

   A-2 Aktor memilih untuk melakukan *delete* data *task*.

   1. Sistem menampilkan data *task* yang akan dihapus.
   2. Aktor melakukan konfirmasi mengenai data *task* yang akan dihapus.

3. Sistem menghapus data *task* yang dimaksud.
4. Berlanjut ke *basic flow* langkah ke-7.

A-3 Aktor memilih untuk melakukan *display* data *task*.

1. Sistem akan menyediakan kategori pencarian.
2. Aktor memasukkan data *task* yang ingin dicari sesuai dengan kriterianya.
3. Sistem melakukan pengecekan terhadap *database task*.

   E-3 Data *task* tidak ditemukan dalam *database*.
4. Sistem menampilkan data *task* sesuai dengan yang ditampilkan.
5. *Use case* selesai.

6. ***Error Flow***

E-1 Data *task* baru yang diinputkan salah.

1. Sistem menampilkan peringatan bahwa data *task* baru yang dimasukkan tidak *valid*.
2. Kembali ke *basic flow* langkah ke-4.

E-2 Data *task* yang telah diedit salah.

1. Sistem menampilkan peringatan bahwa data *task* yang diedit tidak *valid*.
2. Kembali ke *alternatif flow* A-1 langkah ke-2.

E-3 Data *task* tidak ditemukan dalam *database*.

1. Sistem menampilkan informasi bahwa data *branch* yang ingin ditampilkan tidak ada.
2. Kembali ke *alternatif flow* A-3 langkah ke-2.

7. ***PreConditions***
   1. *Use Case Login* telah dilakukan.
   2. Aktor telah memasuki sistem.
8. ***PostConditions***
   1. Data *task* telah ter-*update* di dalam *database*.

4.1.11. *Use Case Spesification* : *Manage Mail*

1. ***Brief Description***

   Use case ini digunakan oleh aktor untuk melakukan pengelolaan *mail* yang akan dilakukan oleh masing-masing aktor. Aktor dapat melakukan *create* dan *delete mail*.

2. ***Primary Actor***
   1. *Admin*
   2. *Customer Care*
   3. Agen
   4. Customer
   5. Central Manager
3. ***Supporting Actor***

   none

4. *Basic Flow*

1. *Use Case* ini dimulai ketika aktor memilih untuk melakukan pengelolaan *mail*.
2. Sistem akan menampilkan antarmuka untuk melakukan pengelolaan *mail*.
3. Aktor memilih untuk melakukan *create mail*.

   A-1 Aktor memilih untuk melakukan *delete mail*.
4. Aktor menuliskan *mail* yang baru.
5. Aktor memilih untuk melakukan pengiriman mail.
6. *Use case* selesai.

5. *Alternatif Flow*

A-1 Aktor memilih untuk melakukan *delete mail*.

1. Sistem menampilkan data *mail* yang akan dihapus.
2. Aktor memilih *mail* yang akan dihapus.
3. Sistem akan menghapus *mail* sesuai pilihan user.
4. Berlanjut ke *basic flow* langkah ke-6.

6. *Error Flow*

none

7. *PreConditions*

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. *PostConditions*

1. Data *mail* telah ter-*update* di dalam *database*.

4.1.12. *Use Case Spesification* : *Manage Password*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk mengubah *password* dari *user* yang bersangkutan.

2. *Primary Actor*

1. *Admin*
2. *Customer Care*
3. Agen
4. Customer
5. Central Manager

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use Case* ini dimulai ketika aktor memilih untuk melakukan pengelolaan *password*.
2. Sistem akan menampilkan antarmuka untuk melakukan pengelolaan *password*.
3. Aktor melakukan perubahan terhadap *password*.
4. Sistem akan menyimpan perubahan yang dilakukan *user*.
5. *Use case* selesai.

5. *Alternatif Flow*

none

6. *Error Flow*

none

7. *PreConditions*

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. *PostConditions*

1. Data *login* telah ter-*update* di dalam *database*.

4.1.13. *Use Case Spesification* : *Agen Report*

1. *Brief Description*

Use case ini digunakan oleh aktor untuk mendapakan laporan atau memperoleh informasi mengenai hal-hal yang berhubungan dengan agen, seperti data pribadi ataupun data yang berkaitan dengan *customer* yang agen tersebut dapatkan.

2. *Primary Actor*

1. *Customer Care*
2. Agen

3. *Supporting Actor*

none

4. *Basic Flow*

1. *Use Case* ini dimulai ketika aktor memilih untuk mendapatkan *Agen Report*.
2. Sistem akan menampilkan antarmuka untuk mendapatkan *Report* sesuai dengan pilihan user.
3. *Use case* selesai.

5. ***Alternatif Flow***

none

6. ***Error Flow***

none

7. ***PreConditions***

1. *Use Case Login* telah dilakukan.
2. Aktor telah memasuki sistem.

8. ***PostConditions***

1. *Report* mengenai Agen dalam *database* akan ditampilkan.

4.1.14. *Use Case Spesification* : *Customer Report*

1. ***Brief Description***

Use case ini digunakan oleh aktor untuk mendapakan laporan atau memperoleh informasi mengenai hal-hal yang berhubungan dengan *customer* yang dia peroleh, misalnya data pribadi.

2. ***Primary Actor***

1. *Customer Care*
2. Agen

3. ***Supporting Actor***

none

4. ***Basic Flow***

1. *Use Case* ini dimulai ketika aktor memilih untuk mendapatkan *Customer Report*.

2. Sistem akan menampilkan antarmuka untuk mendapatkan *Report* sesuai dengan pilihan user.
3. *Use case* selesai.

5. ***Alternatif Flow***

   none

6. ***Error Flow***

   none

7. ***PreConditions***

   1. *Use Case Login* telah dilakukan.
   2. Aktor telah memasuki sistem.

8. ***PostConditions***

   1. *Report* mengenai *Customer* dalam *database* akan ditampilkan.

4.1.15. *Use Case Spesification* : *Product Report*

1. ***Brief Description***

   Use case ini digunakan oleh aktor untuk mendapakan laporan atau memperoleh informasi mengenai perbandingan antara produk yang paling diminati ataupun juga sebaliknya.

2. ***Primary Actor***

   1. *Central Manager*

3. ***Supporting Actor***

   none

4. ***Basic Flow***

   1. *Use Case* ini dimulai ketika aktor memilih untuk mendapatkan *Product Report*.

2. Sistem akan menampilkan antarmuka untuk mendapatkan *Report* sesuai dengan pilihan user.
3. *Use case* selesai.

5. ***Alternatif Flow***

   none

6. ***Error Flow***

   none

7. ***PreConditions***

   1. *Use Case Login* telah dilakukan.
   2. Aktor telah memasuki sistem.

8. ***PostConditions***

   1. *Report* mengenai produk asuransi dalam *database* akan ditampilkan.

4.1.16. *Use Case Spesification* : *All Branch Report*

1. ***Brief Description***

   Use case ini digunakan oleh aktor untuk mendapakan laporan atau mengetahui posisi semua kantor cabang, baik dalam hal keuangan maupun *customer*.

2. ***Primary Actor***

   1. *Central Manager*

3. ***Supporting Actor***

   none

4. ***Basic Flow***

   1. *Use Case* ini dimulai ketika aktor memilih untuk mendapatkan *Product Report*.
   2. Sistem akan menampilkan antarmuka untuk mendapatkan *Report* sesuai dengan pilihan user.
   3. *Use case* selesai.

5. ***Alternatif Flow***

   none

6. ***Error Flow***

   none

7. ***PreConditions***

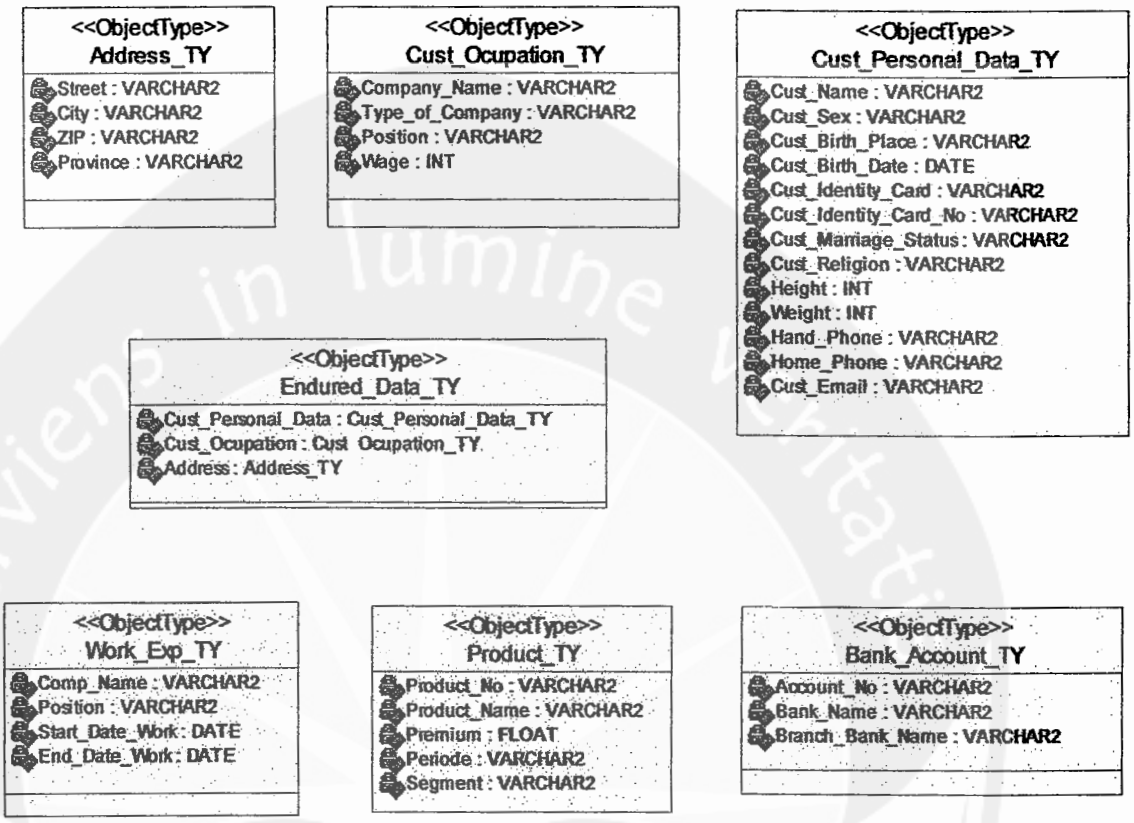
   1. *Use Case Login* telah dilakukan.
   2. Aktor telah memasuki sistem.

8. ***PostConditions***

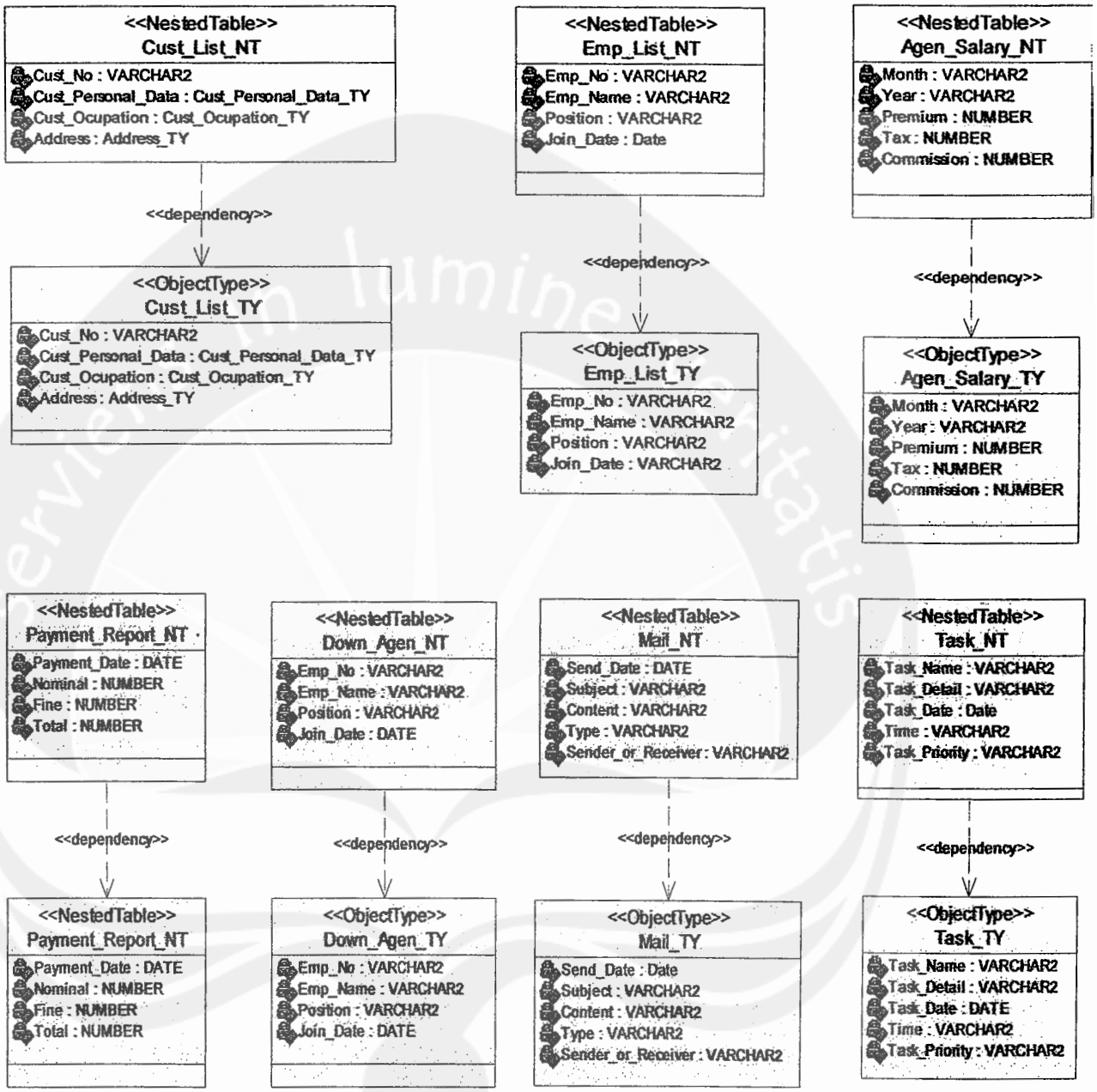
   1. *Report* mengenai semua kantor cabang dalam *database* akan ditampilkan.

4.2. Object-Relational Database

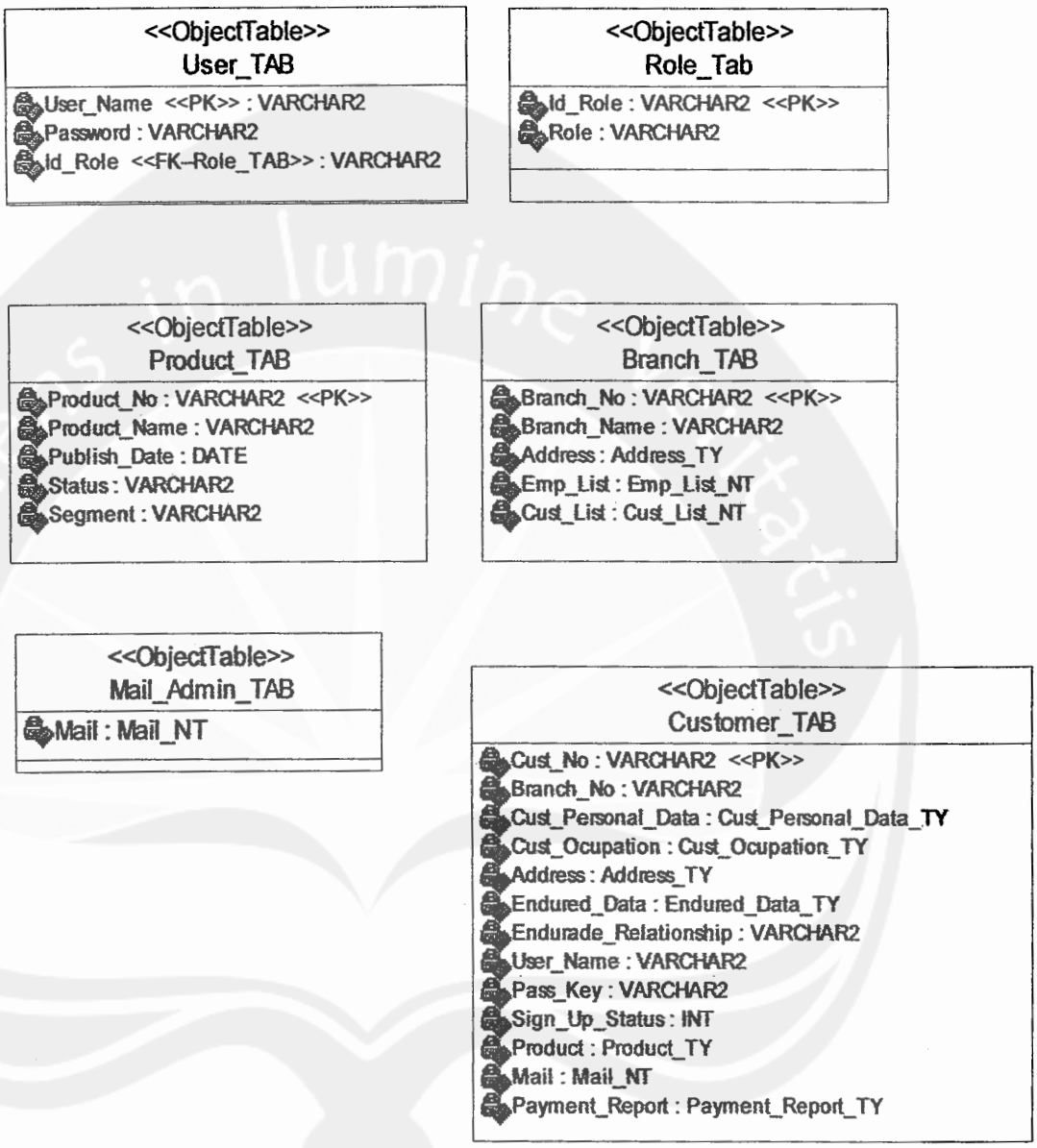
5.1 Object Type

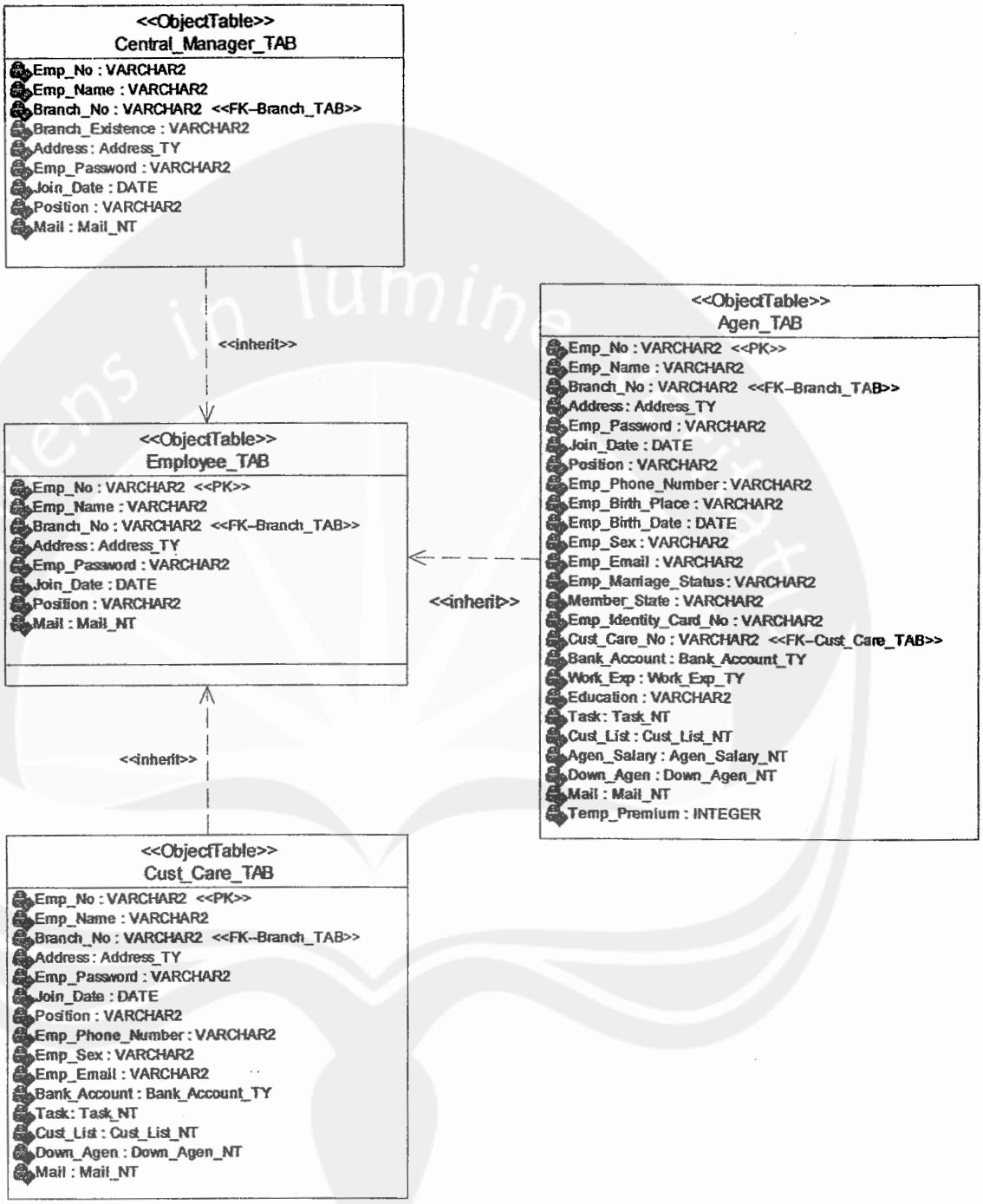


5.2 Object Type dan Nested Table



5.3 Object Table

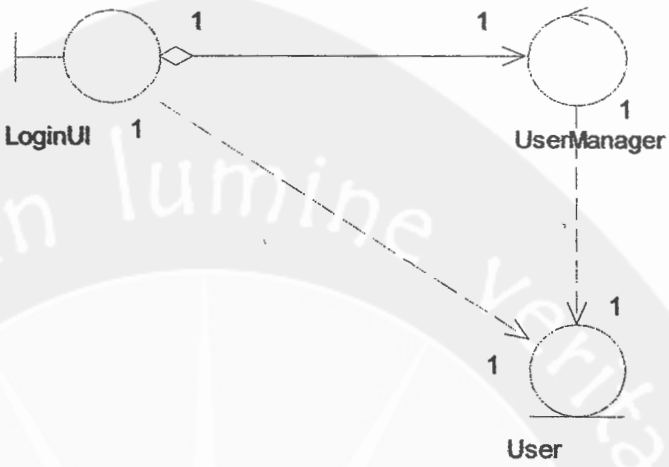




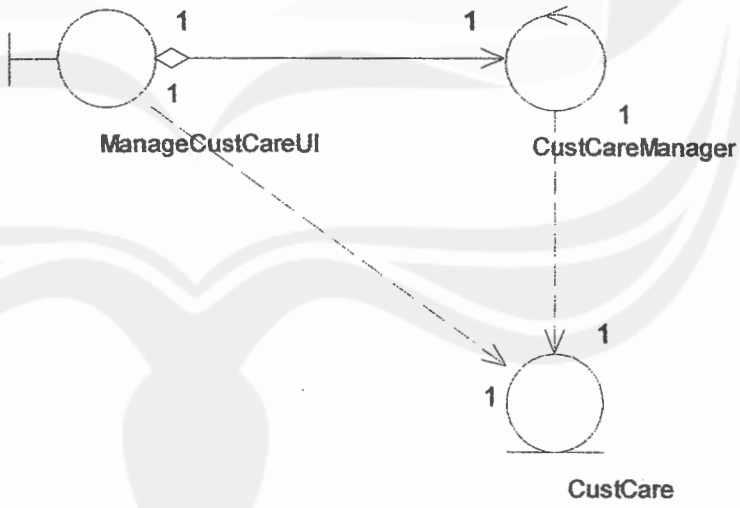
5. Realisasi Use Case

5.1. Static Structure Diagram

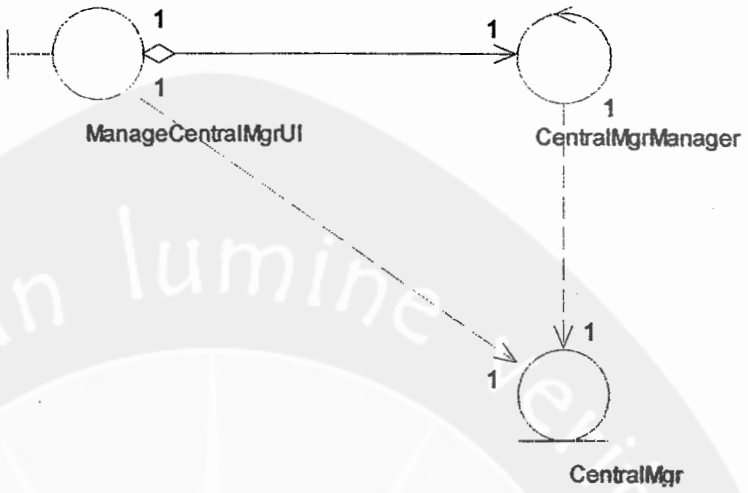
5.1.1. Analysis Class Diagram : Use Case Login



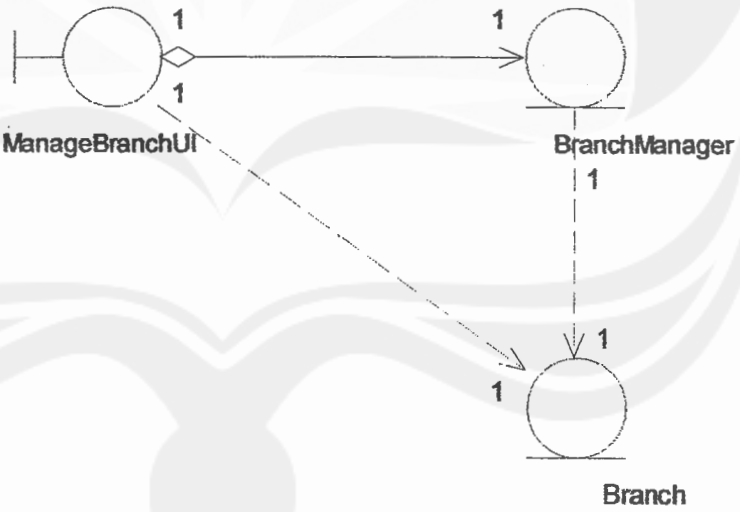
5.1.2. Analysis Class Diagram : Use Case Manage Customer Care



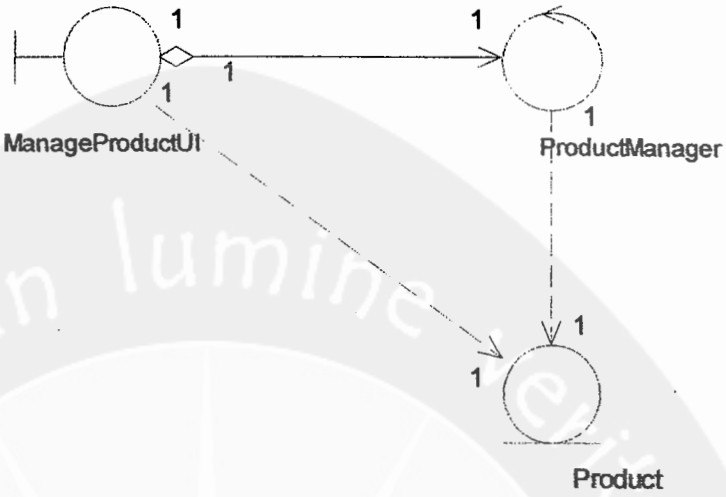
5.1.3. **Analysis Class Diagram** : Use Case Manage Central Manager



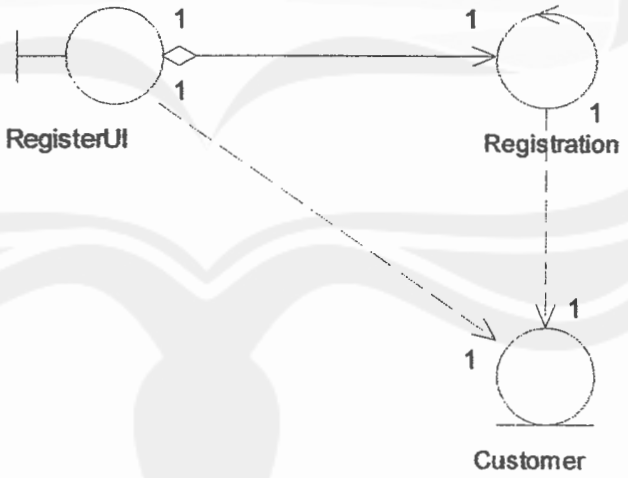
5.1.4. **Analysis Class Diagram** : Use Case Manage Branch



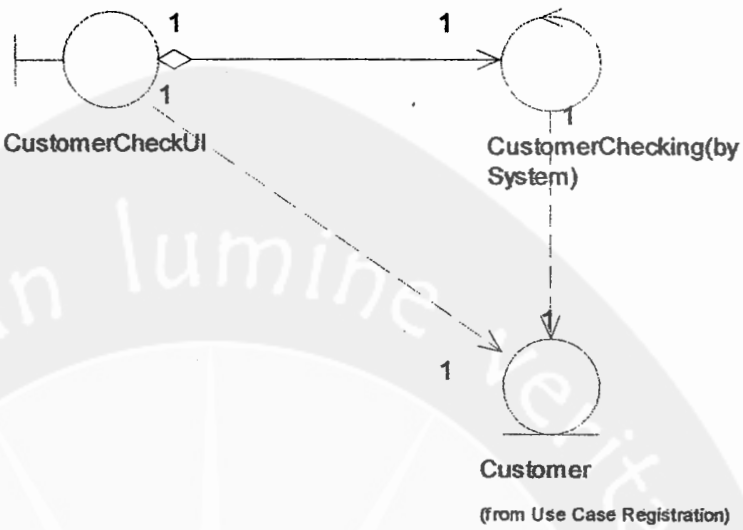
5.1.5. **Analysis Class Diagram** : Use Case Manage Product



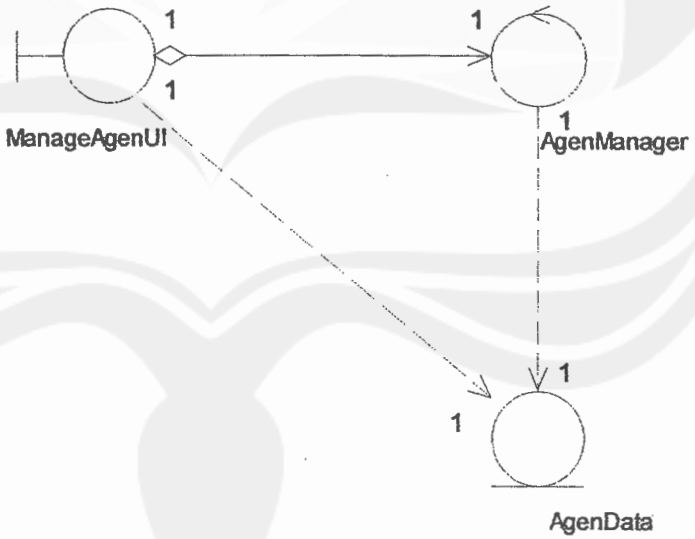
5.1.6. **Analysis Class Diagram** : Use Case Registration



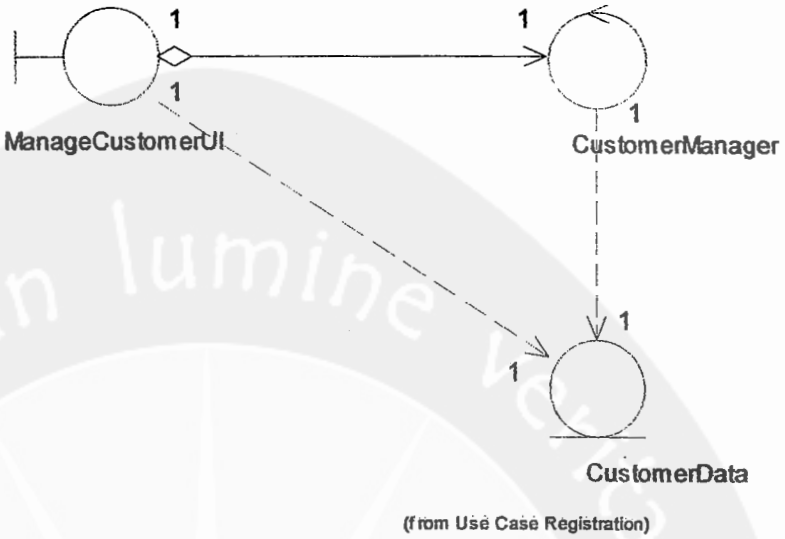
5.1.7. **Analysis Class Diagram** : Use Case Customer Check



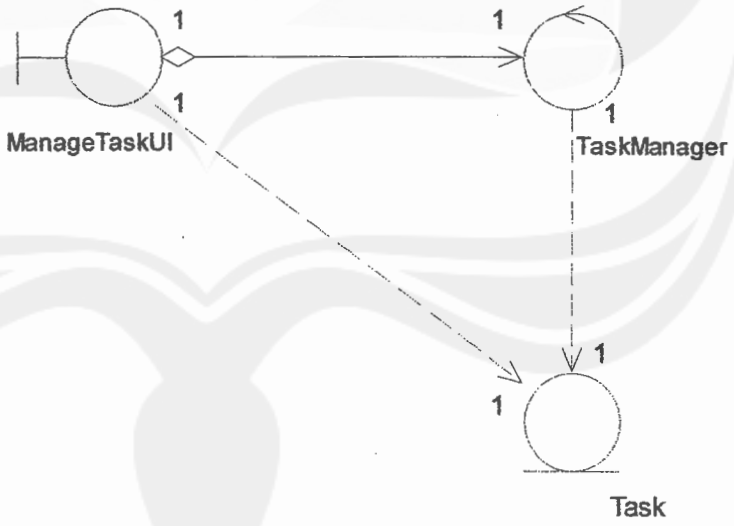
5.1.8. **Analysis Class Diagram** : Use Case Manage Agen



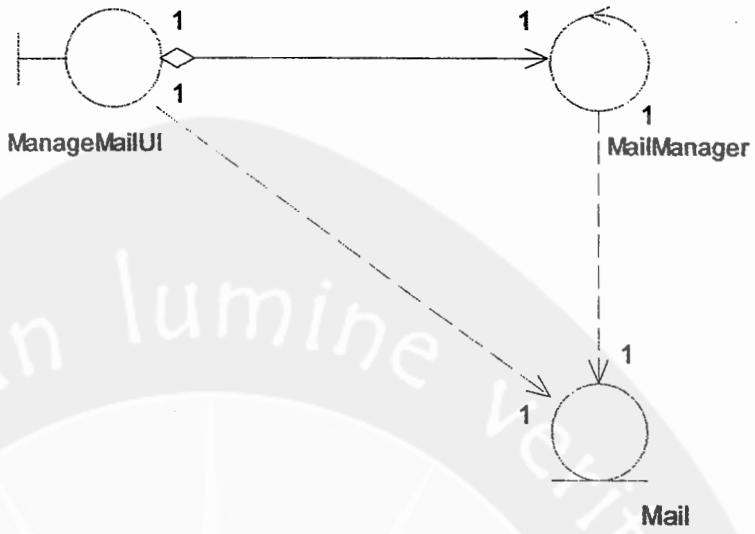
5.1.9. **Analysis Class Diagram** : Use Case Manage Customer



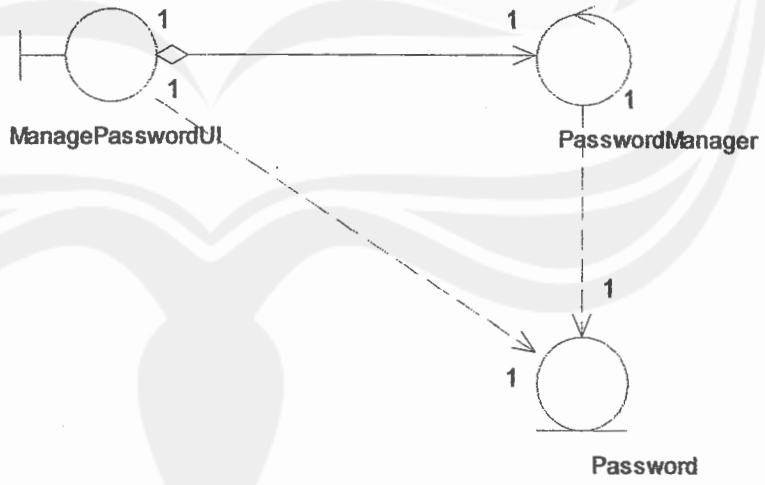
5.1.10. **Analysis Class Diagram** : Use Case Manage Task



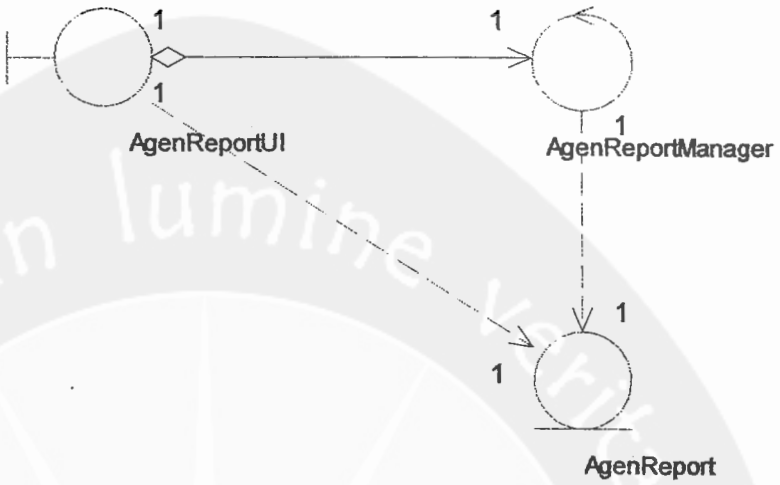
5.1.11. **Analysis Class Diagram** : Use Case Manage Mail



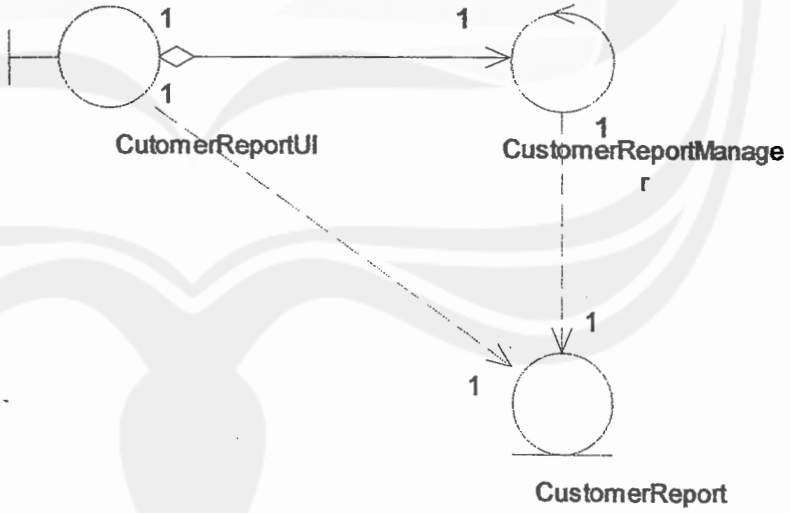
5.1.12. **Analysis Class Diagram** : Use Case Manage Password



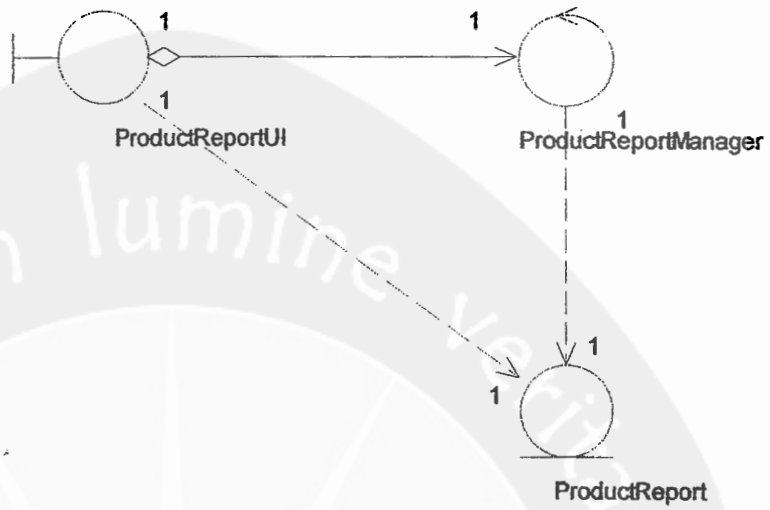
5.1.13. **Analysis Class Diagram** : Use Case Agen Report



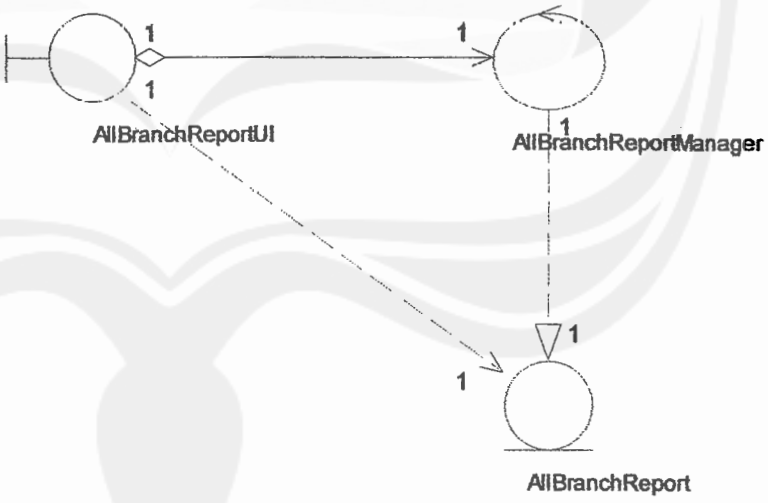
5.1.14. **Analysis Class Diagram** : Use Case Customer Report



5.1.15. **Analysis Class Diagram** : Use Case Product Report

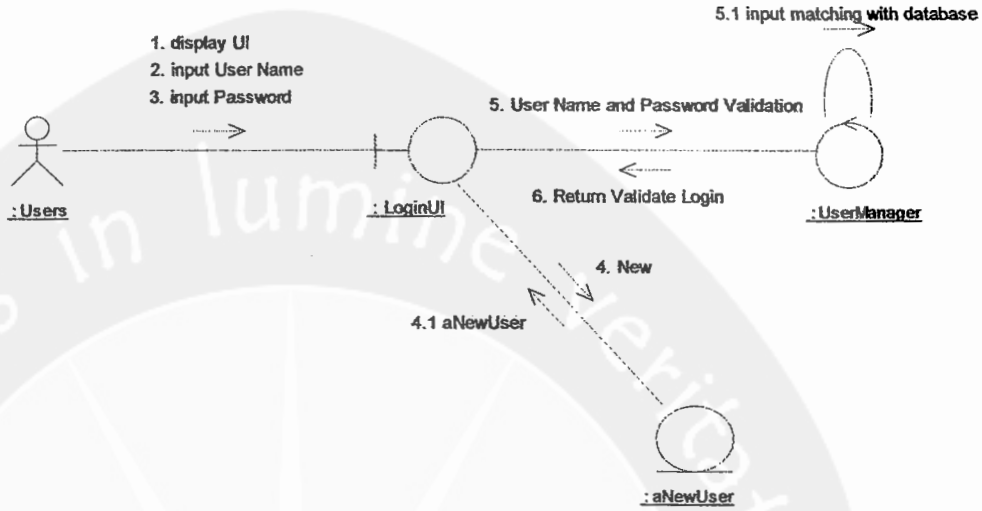


5.1.16. **Analysis Class Diagram** : Use Case All Branch Report



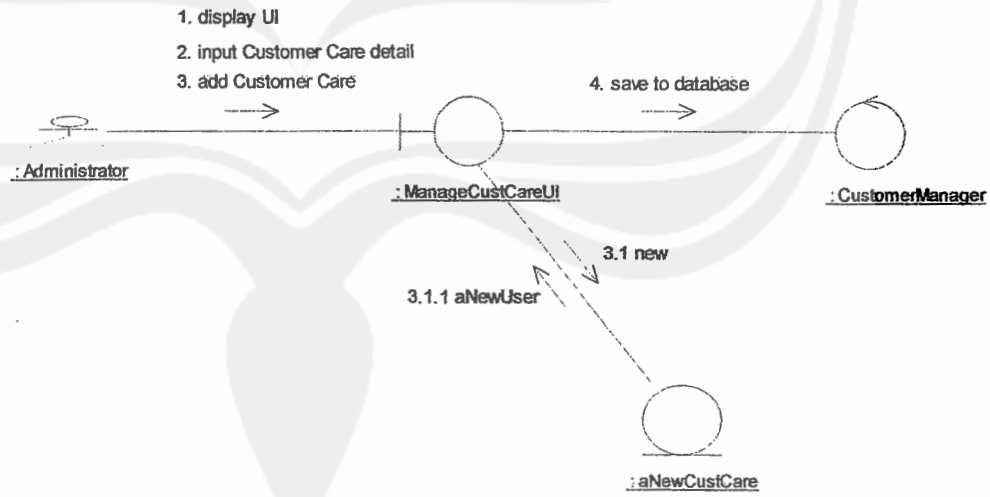
5.2. Interaction Diagram

5.2.1. Analysis Collaboration Diagram : Use Case Login

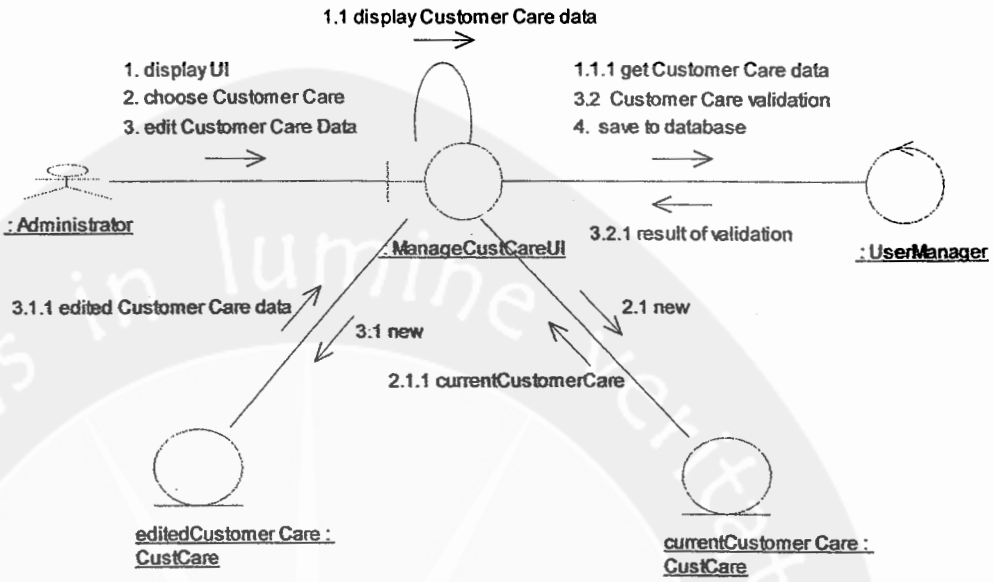


5.2.2. Analysis Collaboration Diagram : Use Case Manage Customer Care

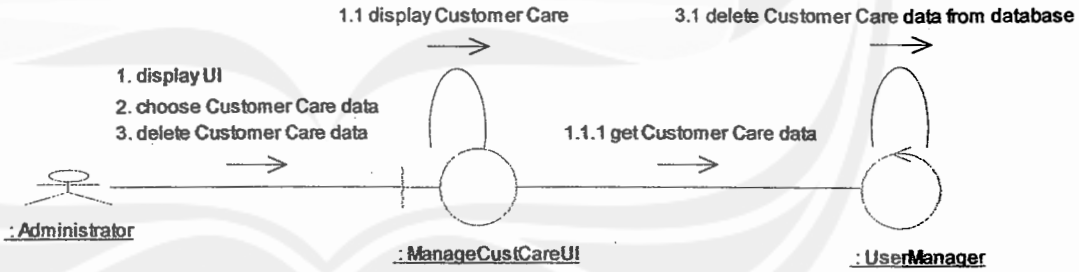
5.2.2.1. Analysis Collaboration Diagram : Use Case Add Customer Care



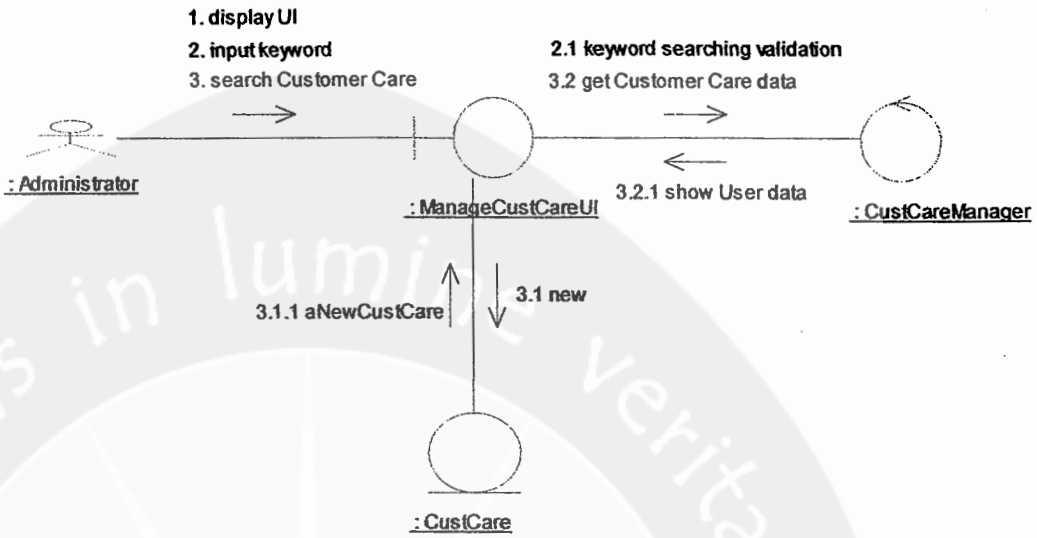
5.2.2.2. Analysis Collaboration Diagram : Use Case Edit Customer Care



5.2.2.3. Analysis Collaboration Diagram : Use Case Delete Customer Care

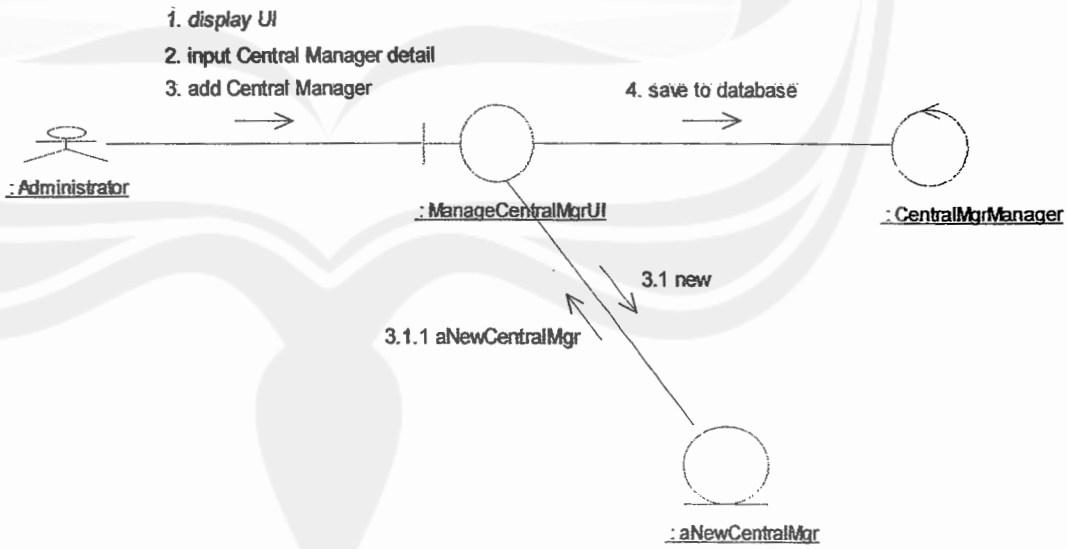


5.2.2.4. Analysis Collaboration Diagram : Use Case Search Customer Care

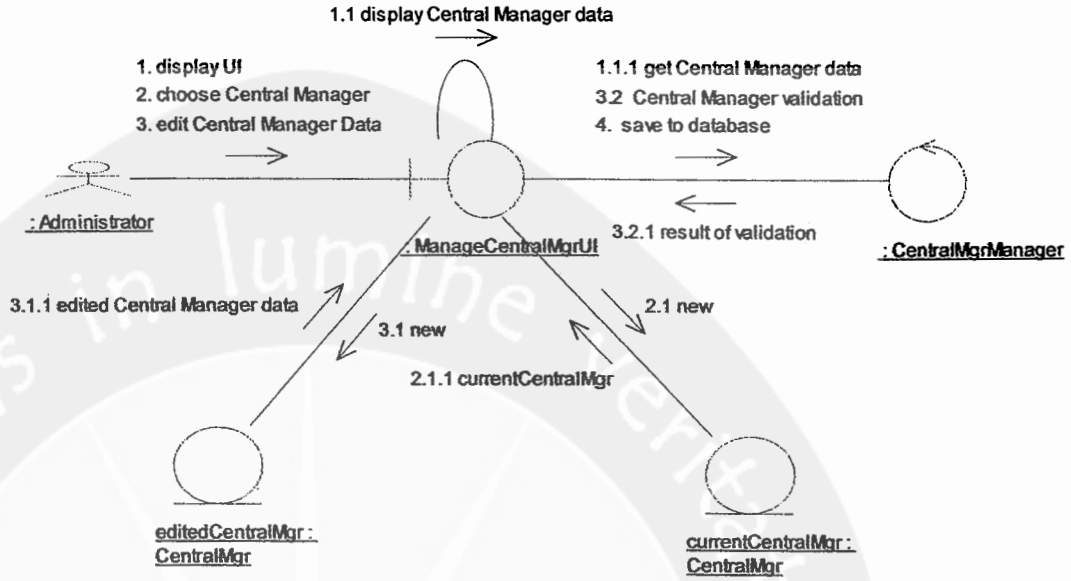


5.2.3. Analysis Collaboration Diagram : Use Case Manage Central Manager

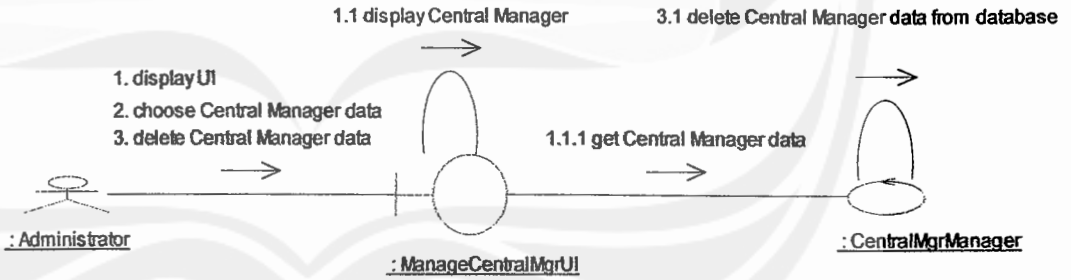
5.2.3.1. Analysis Collaboration Diagram : Use Case Add Central Manager



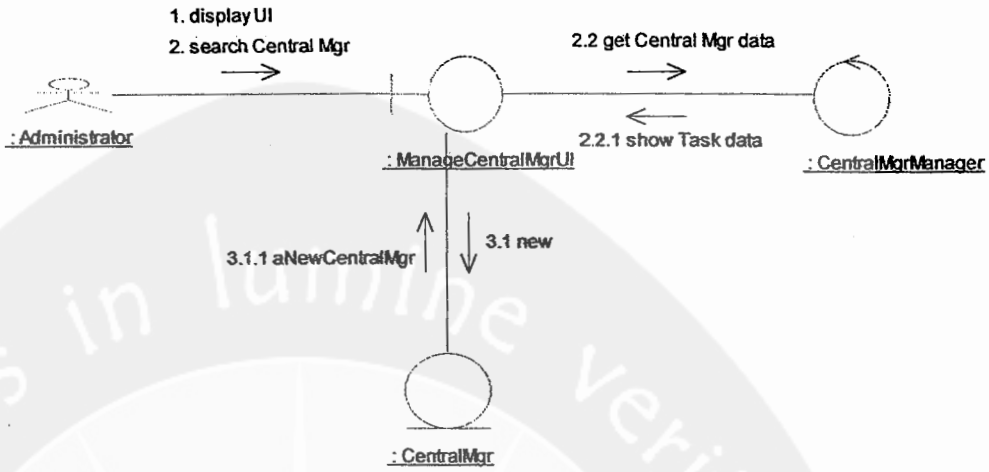
5.2.3.2. Analysis Collaboration Diagram : Use Case Edit Central Manager



5.2.3.3. Analysis Collaboration Diagram : Use Case Delete Central Manager

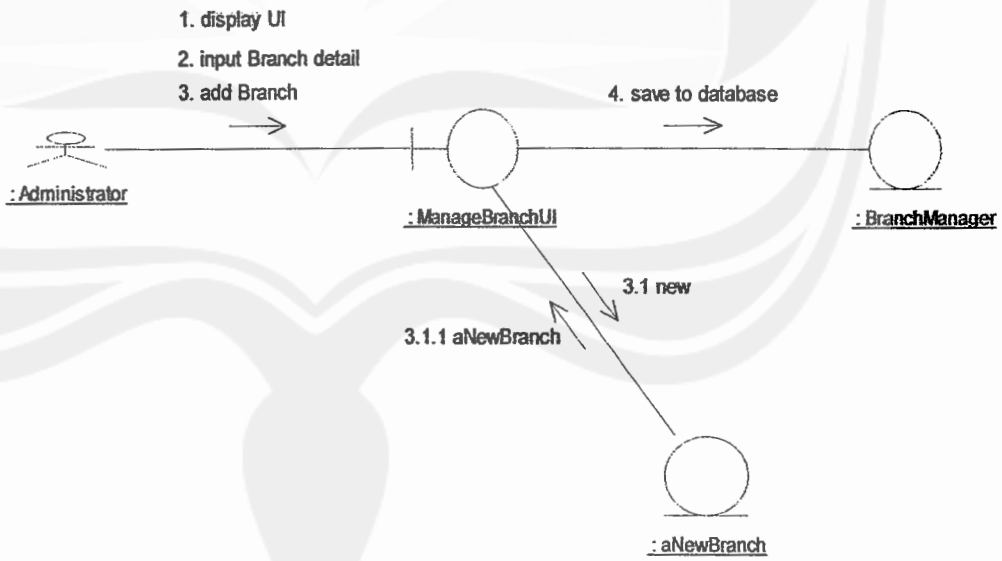


5.2.3.4. Analysis Collaboration Diagram : Use Case Display Central Manager

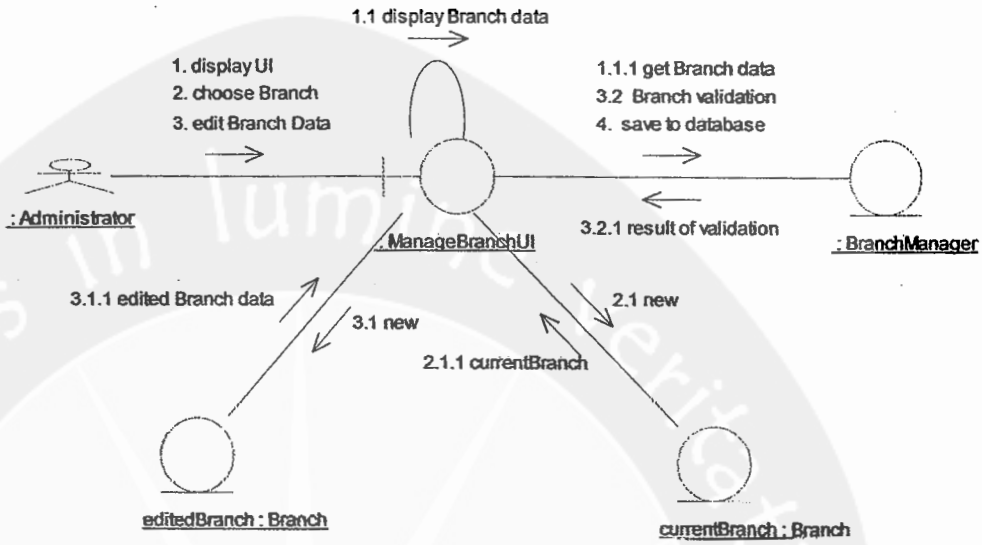


5.2.4. Analysis Collaboration Diagram : Use Case Manage Branch

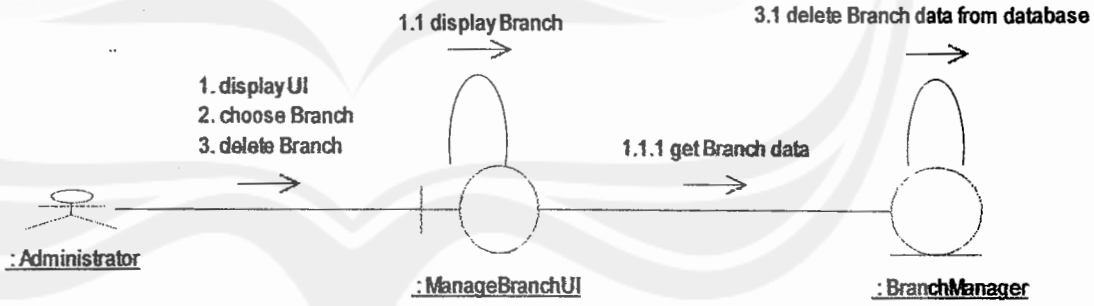
5.2.4.1. Analysis Collaboration Diagram : Use Case Add Branch



5.2.4.2. Analysis Collaboration Diagram : Use Case Edit Branch

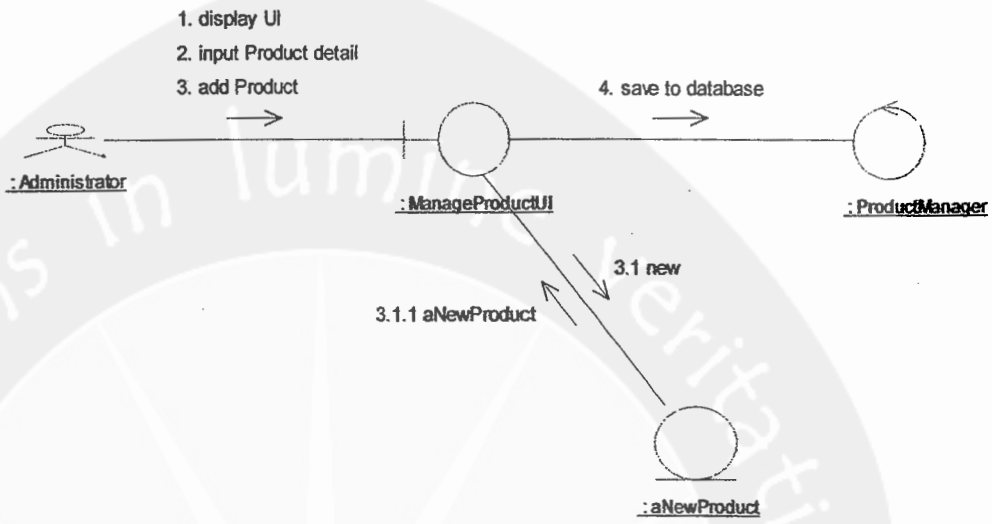


5.2.4.3. Analysis Collaboration Diagram : Use Case Delete Branch

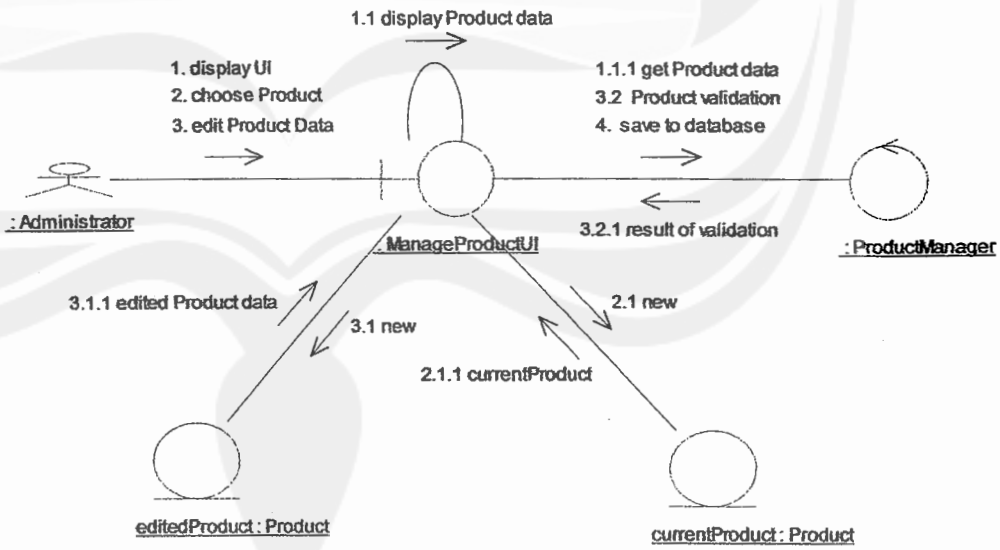


5.2.5. **Analysis Collaboration Diagram** : Use Case Manage Product

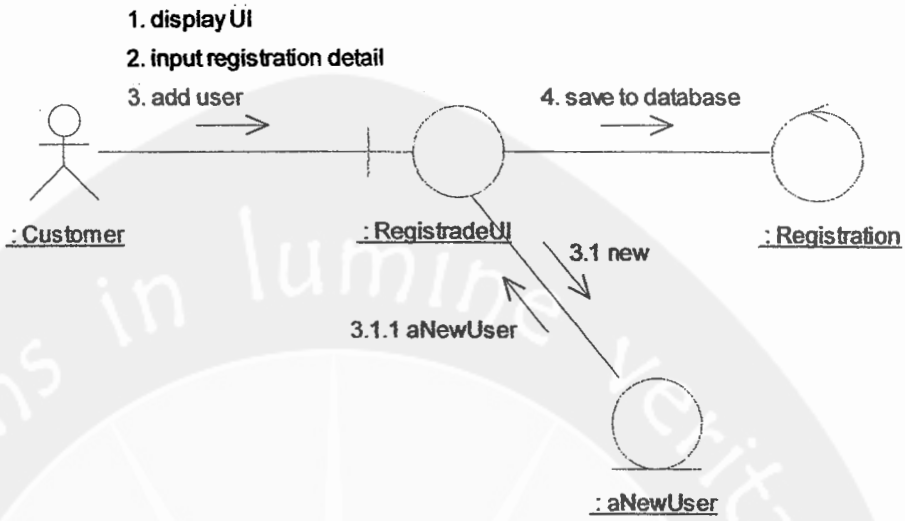
5.2.5.1. **Analysis Collaboration Diagram** : Use Case Add Product



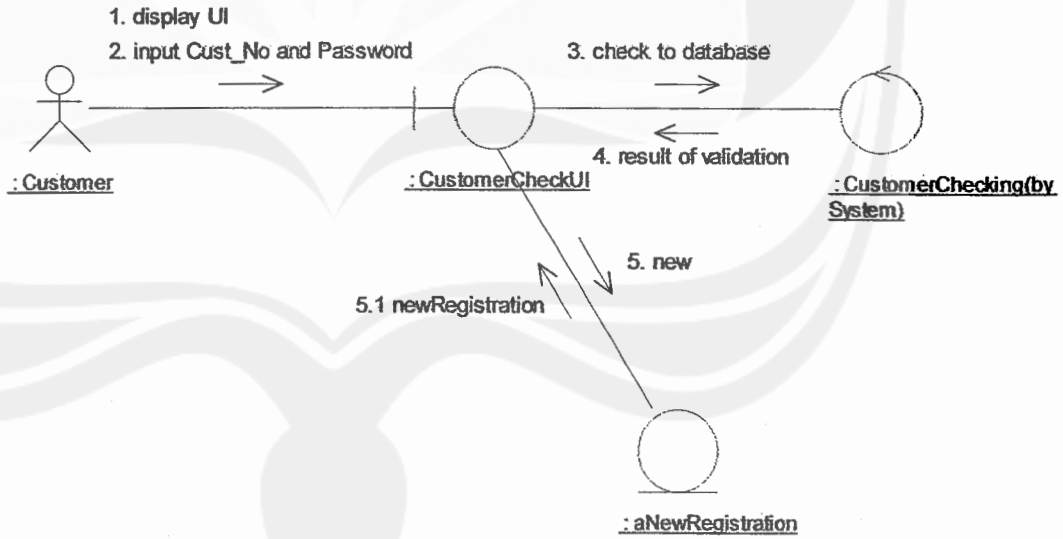
5.2.5.2. **Analysis Collaboration Diagram** : Use Case Edit Product



5.2.6. **Analysis Collaboration Diagram** : Use Case Registration

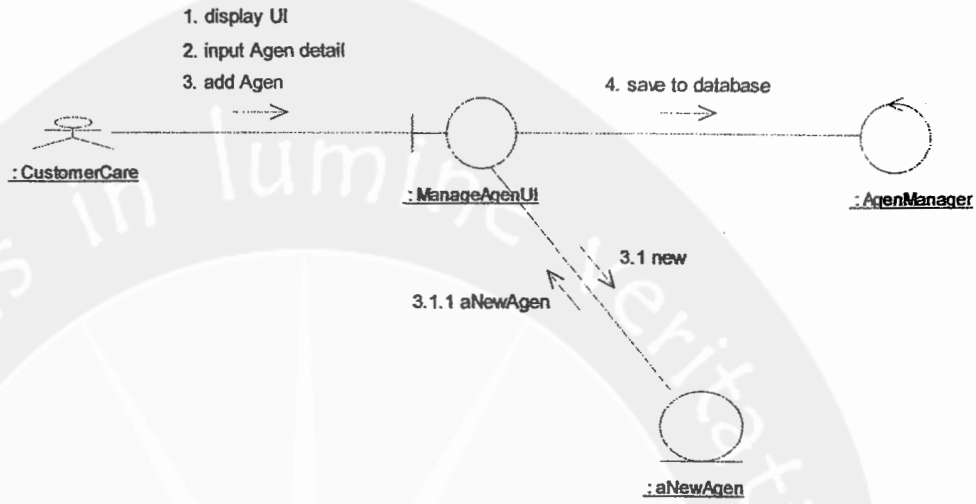


5.2.7. **Analysis Collaboration Diagram** : Use Case Customer Check

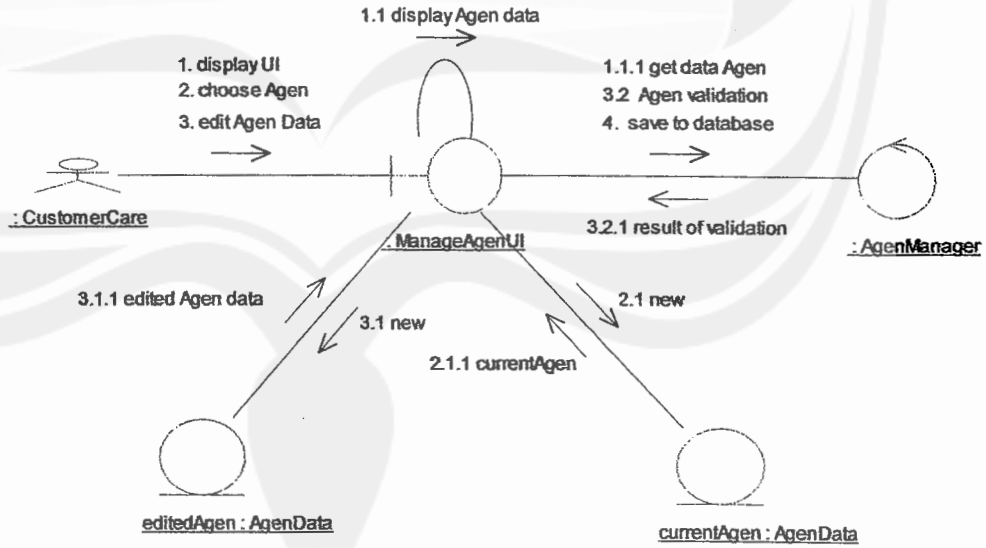


5.2.8. Analysis Collaboration Diagram : Use Case Manage Agen

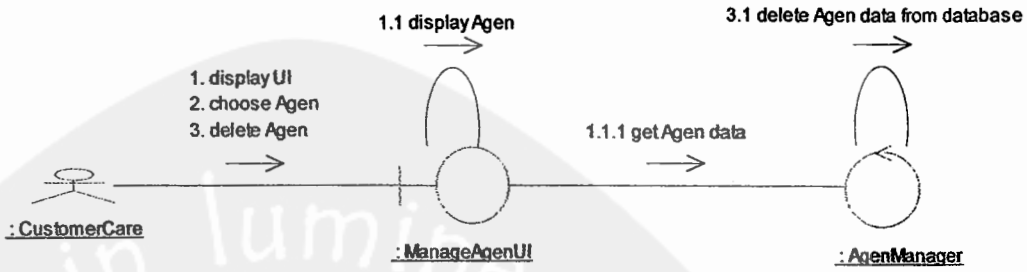
5.2.8.1. Analysis Collaboration Diagram : Use Case Add Agen



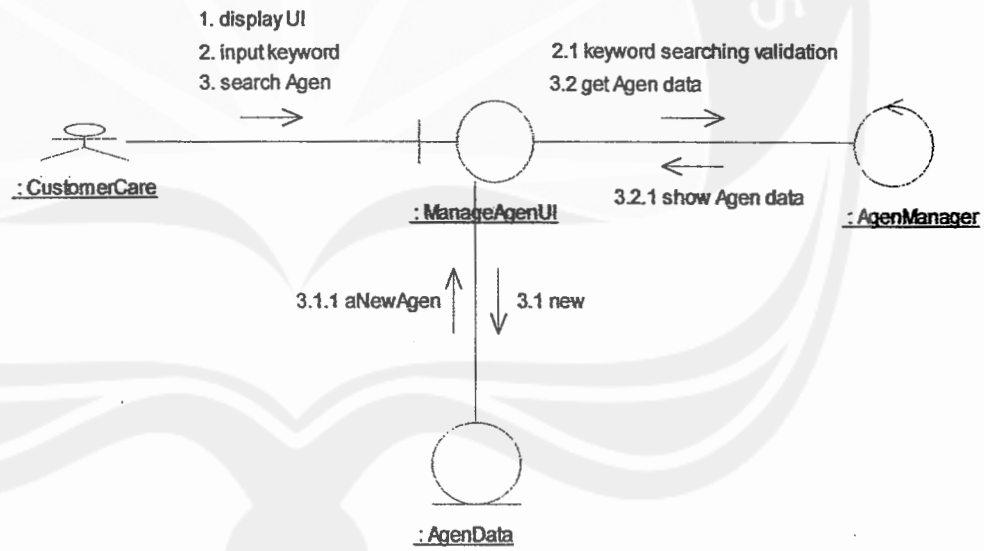
5.2.8.2. Analysis Collaboration Diagram : Use Case Edit Agen



5.2.8.3. Analysis Collaboration Diagram : Use Case Delete Agen

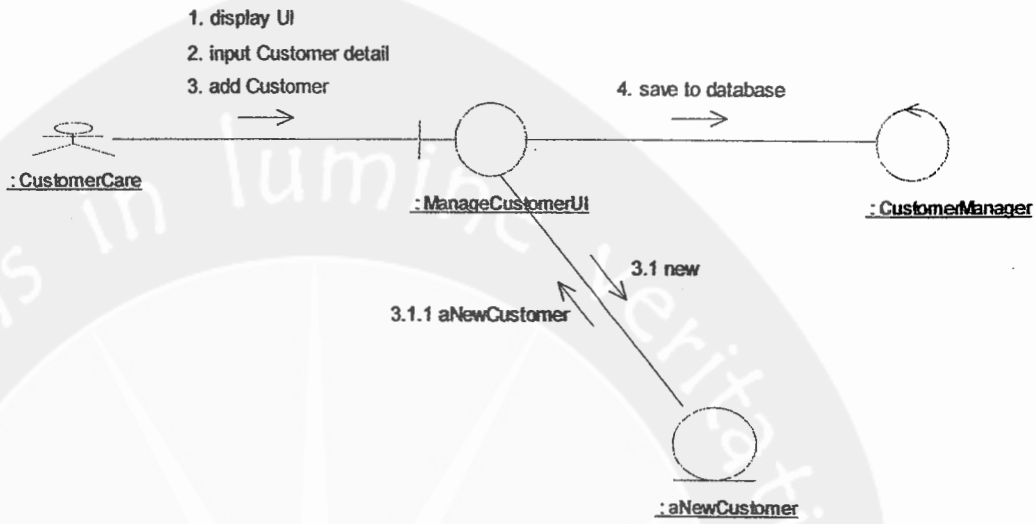


5.2.8.4. Analysis Collaboration Diagram : Use Case Search Agen

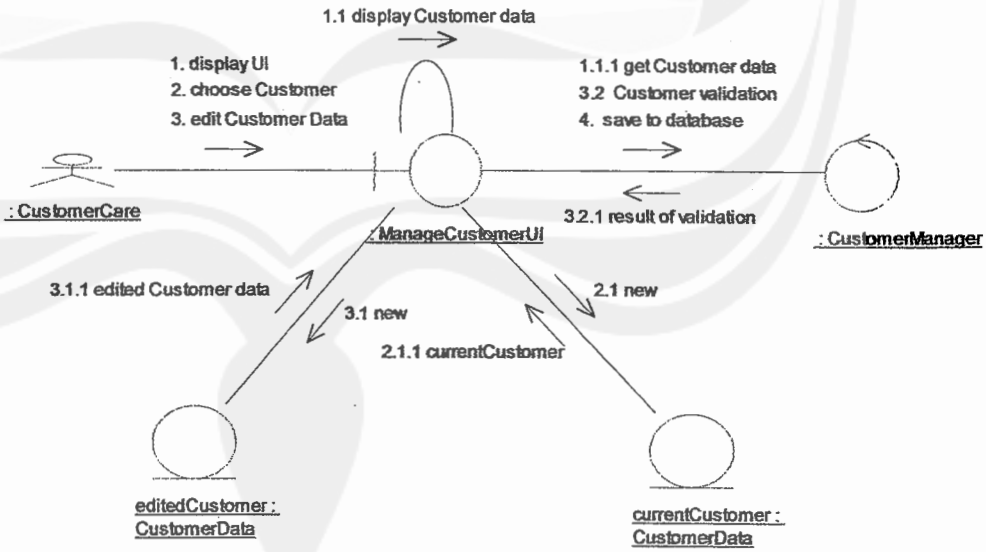


5.2.9. **Analysis Collaboration Diagram** : Use Case Manage Customer

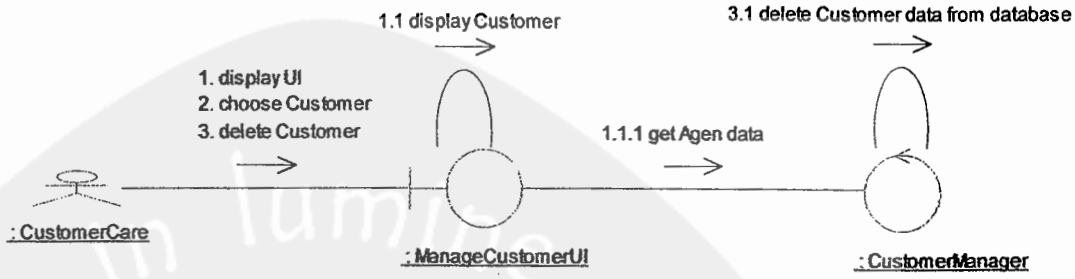
5.2.9.1. **Analysis Collaboration Diagram** : Use Case Add Customer



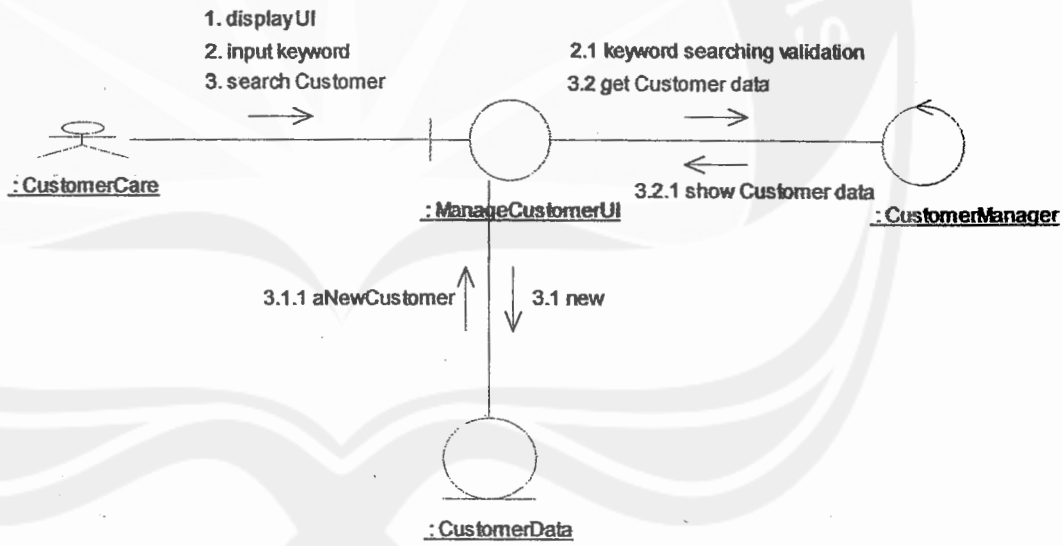
5.2.9.2. **Analysis Collaboration Diagram** : Use Case Edit Customer



5.2.9.3. Analysis Collaboration Diagram : Use Case Delete Customer

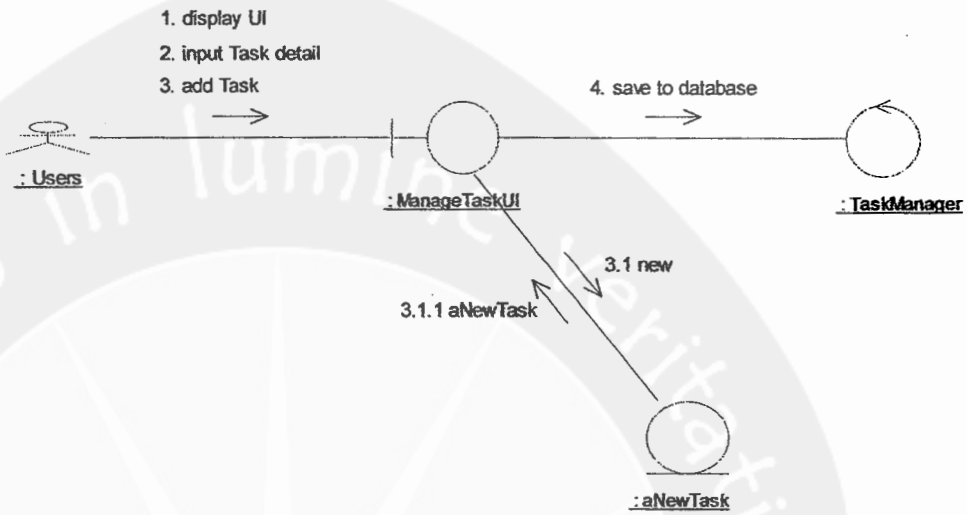


5.2.9.4. Analysis Collaboration Diagram : Use Case Search Customer

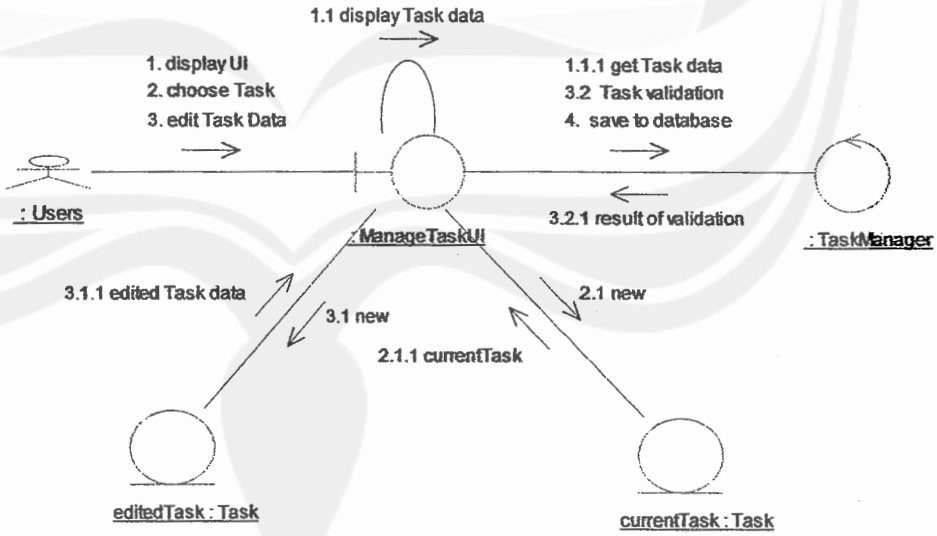


5.2.10. Analysis Collaboration Diagram : Use Case Manage Task

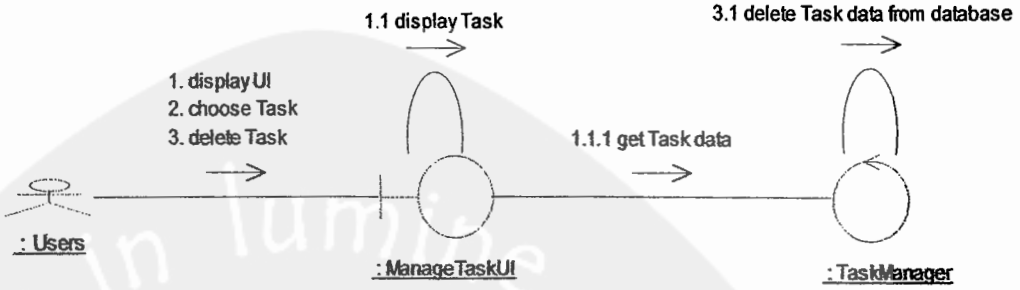
5.2.10.1. Analysis Collaboration Diagram : Use Case Add Task



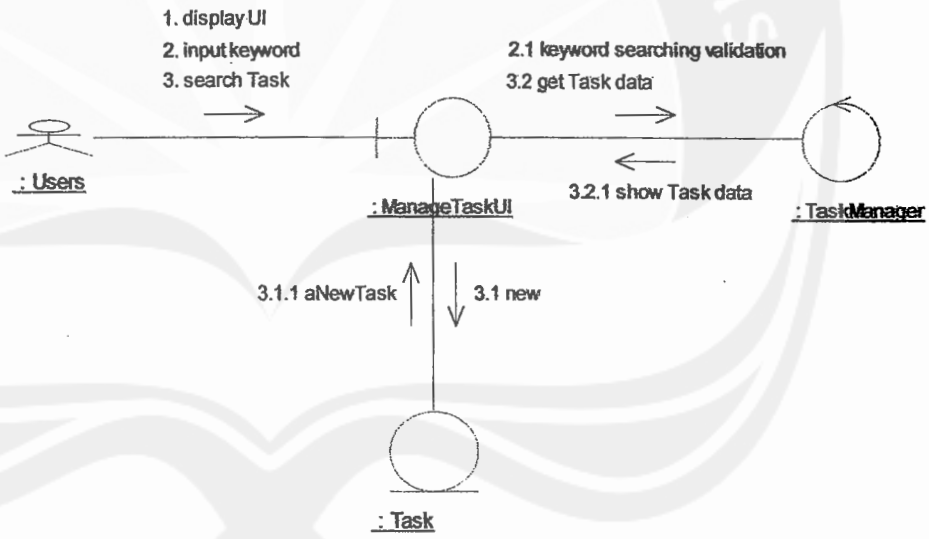
5.2.10.2. Analysis Collaboration Diagram : Use Case Edit Task



5.2.10.3. Analysis Collaboration Diagram : Use Case Delete Task

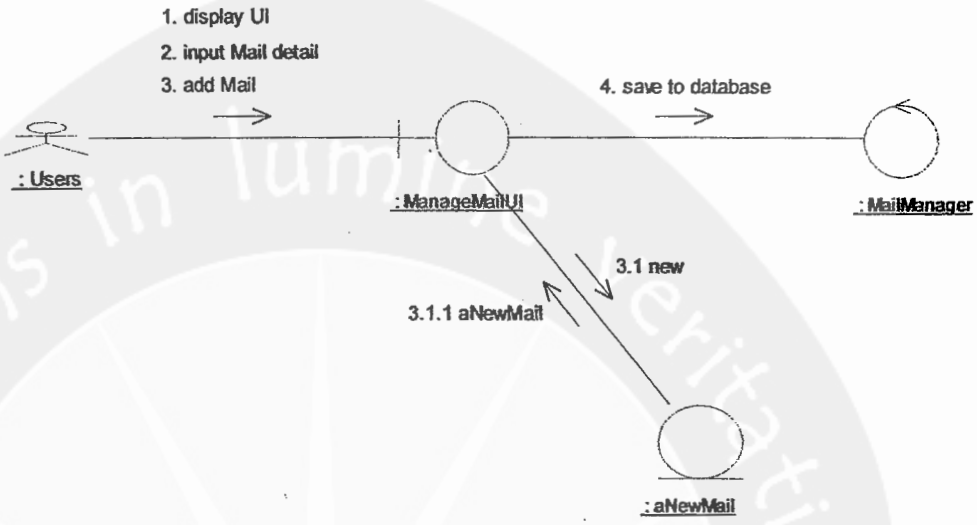


5.2.10.4. Analysis Collaboration Diagram : Use Case Display Task

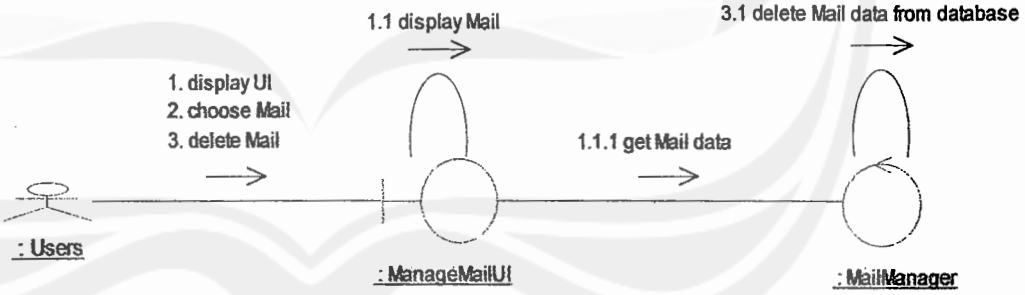


5.2.11. Analysis Collaboration Diagram : Use Case Manage Mail

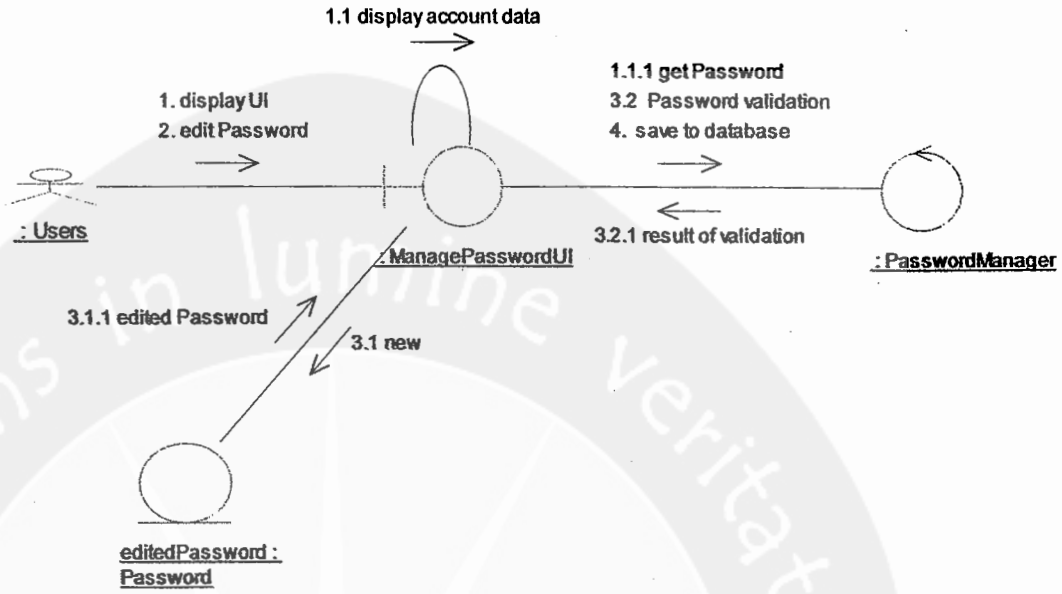
5.2.11.1. Analysis Collaboration Diagram : Use Case Create Mail



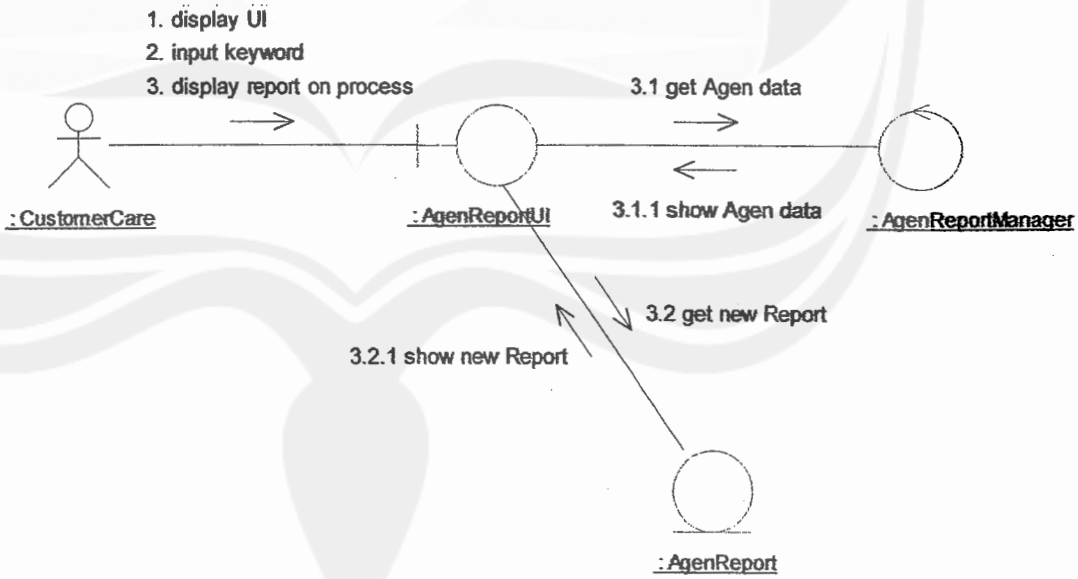
5.2.11.2. Analysis Collaboration Diagram : Use Case Delete Mail



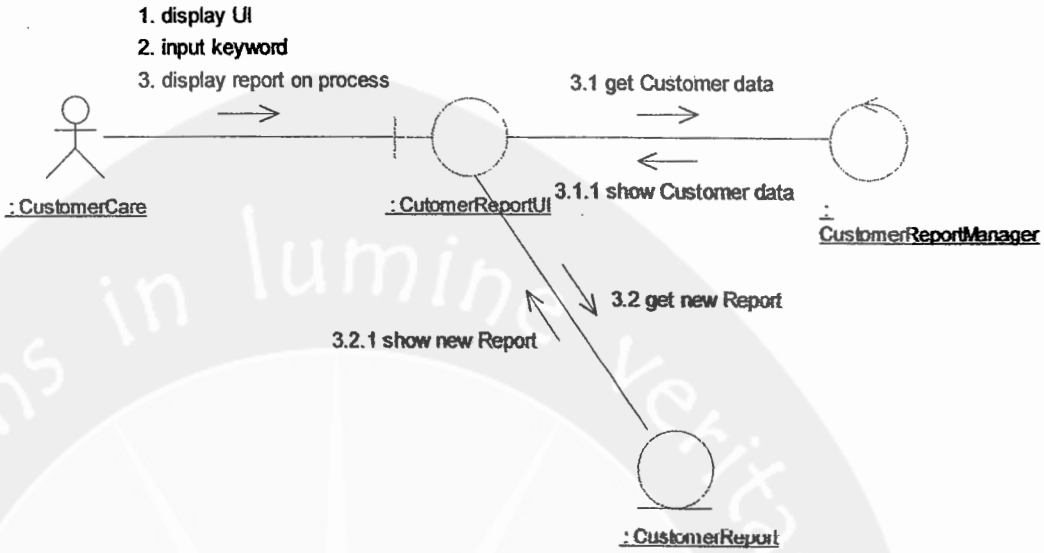
5.2.12. Analysis Collaboration Diagram : Use Case Password Management



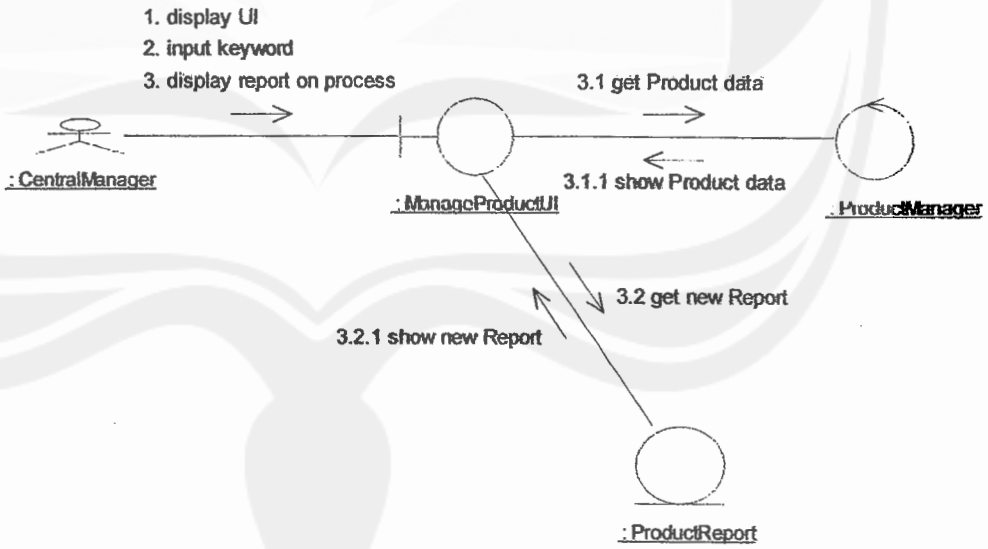
5.2.13. Analysis Collaboration Diagram : Use Case Agen Report



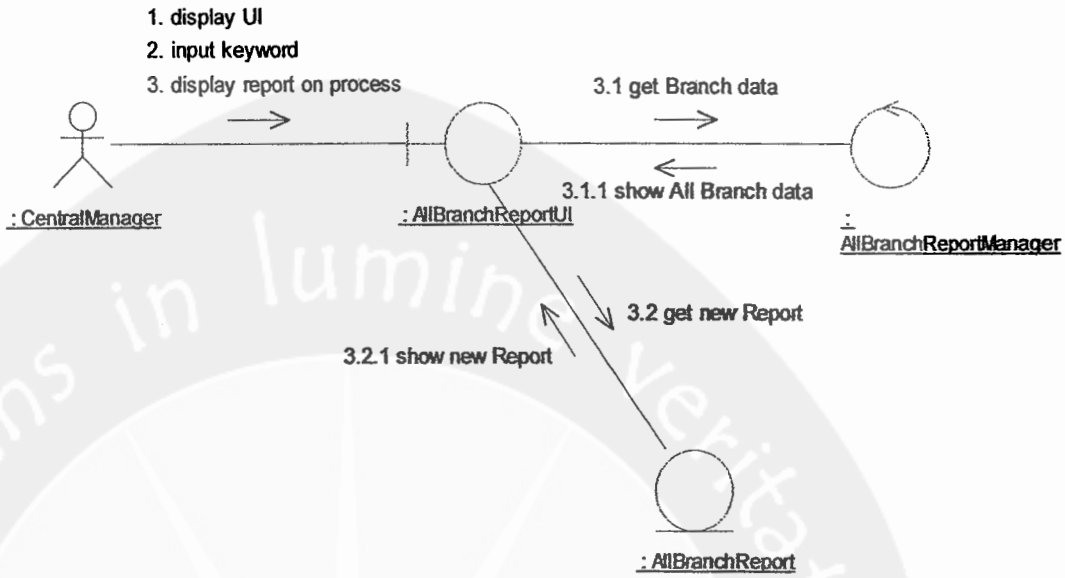
5.2.14. Analysis Collaboration Diagram : Use Case Customer Report



5.2.15. Analysis Collaboration Diagram : Use Case Product Report



5.2.16. Analysis Collaboration Diagram : Use Case All Branch Report



DPPL

DESKRIPSI PERANCANGAN PERANGKAT LUNAK

APLIKASI *CRM* (*CUSTOMER RELATIONSHIP MANAGEMENT*) PADA PERUSAHAAN ASURANSI DENGAN *OBJECT RELATIONAL DATABASE MANAGEMENT SYSTEM*

Dipersiapkan oleh:

ANGGISESA JALULAGA / 3712

Program Studi Teknik Informatika

Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
A	
B	
C	
D	
E	
F	

INDEX TGL	20-6-07	20-7-07					
Ditulis oleh	AJ	AJ					
Diperiksa oleh	IW dan YSP	IW,YSP ,KA, EDU					
Disetujui oleh	IW dan YSP	IW,YSP ,KA, EDU					

Daftar Halaman Perubahan

Halaman	Revisi	Halaman	Revisi

Daftar Isi

1. Pendahuluan

1.1. Tujuan

Tujuan dari Dokumen Deskripsi Perancangan Perangkat Lunak (DPPL) dalam pengembangan perangkat lunak *CRM* (*Customer Relationship Management*) yaitu mendefinisikan perancangan perangkat lunak yang akan dikembangkan. Dokumen DPPL ini akan digunakan oleh pengembang perangkat lunak sebagai acuan untuk implementasi pada tahap selanjutnya. Secara lebih lanjut, DPPL juga merupakan suatu bentuk alat yang digunakan oleh pihak *developer* (pengembang) dan *user* (pengguna) untuk berkomunikasi demi tercapainya suatu pemahaman yang sama terhadap penyusun dasar dari sebuah sistem informasi yang akan dikembangkan. Hal ini akan mempermudah dalam pembelajaran dan pengembangan terhadap perangkat lunak (Software) yang bersangkutan.

Di dalam DPPL ini akan dibahas mengenai deskripsi dari entitas-entitas yang ada (semua antitas yang berhubungan dengan perangkat lunak), rancangan arsitektur (gambaran form per form yang dapat digunakan oleh user), serta perancangan antarmuka dan fungsionalitasnya (deskripsi maksud dan proses yang terjadi).

1.2. Lingkup Masalah

Perangkat Lunak *Customer Relationship Management* ini merupakan perangkat lunak yang mengambil *business process* dari PT. Asuransi

Allianz Life Indonesia dengan tujuan membangun sebuah perangkat lunak yang mampu membantu dalam melakukan pengelolaan atau manajemen untuk membina hubungan yang baik dengan klien/*customer*, yang meliputi pencatatan, penyimpanan, pencarian, serta memberikan laporan yang aktual dan informatif terhadap data-data yang berhubungan dengan relasi antara perusahaan dan *customer*. Data-data tersebut meliputi:

1. Data user, merupakan pengguna dari perangkat lunak *CRM* ini.
2. Data produk asuransi, merupakan jenis-jenis fasilitas asuransi yang ditawarkan.
3. Data customer, merupakan pengguna atau anggota dari produk asuransi yang ditawarkan.
4. Data *employee*, merupakan pegawai pada perusahaan. *Employee* nantinya akan dibagi menjadi 3 jenis, yaitu:
   a. *Customer care*, merupakan pihak yang menjaga hubungan antara perusahaan dengan *customer* dan juga bertindak sebagai *Branch Manager* pada perusahaan yang bersangkutan.
   b. Agen (marketing/sales), merupakan pihak yang bertugas mencari customer dan merupakan bawahan dari Customer Care.
   c. Central *Manager*, adalah *employee* yang membawahi semua kantor cabang perusahaan.
5. Data tugas (task), merupakan jadwal harian untuk agen dan customer care.

6. Data event dan iklan, merupakan acara-acara dan iklan-iklan yang diadakan atau dibuat perusahaan untuk tujuan promosi.

Berhubungan dengan data-data yang ditampilkan di atas, maka tujuan dari pengembangan perangkat lunak *Customer Relationship Management* ini adalah menangani pengelolaan data-data sebagai berikut:

1. Menangani **pengelolaan *user***, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta memberikan hak akses kepada masing-masing *user* yang menggunakan perangkat lunak.
2. Menangani **pengelolaan produk asuransi**, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta hubungannya dengan customer.
3. Menangani pengelolaan customer, digunakan untuk melakukan fungsi tambah, edit, hapus, cari, serta mengelola hubungannya dengan customer care dan agen(sales).
4. Menangani pengelolaan employee, digunakan untuk melakukan fungsi tambah, edit, hapus, dan cari terhadap data employee/pegawai. Employee sendiri akan dibagi menjadi 3 jenis, yaitu Agen, Customer Care dan Central Manager.
5. Menangani pengelolaan tugas/task. Digunakan untuk melakukan fungsi tambah, edit, hapus, serta manajemen terhadap daftar tugas yang diberikan dari pihak customer care terhadap agen.

6. Menangani pengelolaan mail. Digunakan untuk melakukan fungsi tambah, edit, hapus, dan proses pengirimannya dari satu user ke user yang lain.
7. Menangani pengelolaan report, digunakan untuk memperoleh informasi yang berhubungan dengan data-data dan pengelolaan yang telah dipapar sebelumnya.

1.3. Definisi Akronim dan Singkatan

Daftar definisi akronim dan singkatan:

Keyword/Phrase	Definisi
Agen	Pihak perusahaan yang bertugas untuk mencari dan menjaga *customer*.
Branch	Kantor cabang perusahaan.
Competitor	Perusahaan pesaing yang bergerak di bidang yang sama (asuransi).
CRM(Customer Relationsip Management)	Perangkat lunak yang digunakan untuk membantu dalam pengelolaan/manajemen terhadap relasi dengan *customer*.
Customer	Pengguna atau anggota yang terdaftar di dalam salah satu jenis fasilitas asuransi yang ditawarkan.
Customer Care	Pihak perusahaan yang bertugas untuk menjaga hubungan dengan *customer*.
DataBase	Merupakan kelompok data (arsip) yang saling berhubungan dan diorganisir sedemikianrupa agar dapat menghasilkan informasi dan dapat dimanfaatkan kembali dengan cepat dan

	mudah.
ERD	*Entity Relationship Diagram* merupakan teknis grafis/diagram yang menggambarkan *objek* dan hubungan antar *objek*.
Event	Acara-acara yang diadakan oleh perusahaan untuk tujuan promosi.
Hak Akses	Hak yang dimiliki seorang *user* untuk menggunakan sistem yang diatur oleh *administrator* (dalam kasus ini bagian *Supervisor*).
Internet	Internet merupakan istilah umum yang dipakai untuk menunjuk *Network* global yang terdiri dari komputer dan layanan servis dengan sekitar 30 sampai 50 juta pemakai komputer dan puluhan layanan informasi termasuk e-mail, FTP, dan World Wide Web.
Login	Sebuah ketentuan (berupa *interface*) yang akan menyeleksi hak akses dari seorang *user* (kesesuaian antara *user name* dengan *password-nya*).
ORDBMS	*Object Relational DataBase Management System* atau sistem pengelolaan untuk *database* dengan konsep relasi antar *object*.

Premi	Uang iuran yang dibayarkan kepada pihak perusahaan asuransi untuk memenuhi kewajiban *customer*.
Prospect Customer	*Customer* yang sedang diusahakan oleh agen untuk *closing* (menjadi pengguna atau anggota produk asuransi yang ditawarkan).
Report	Laporan yang akan disajikan kepada seseorang atau pihak sesuai dengan kebutuhannya (berdasar hak aksesnya).
Server	Komputer yang menyediakan sumber daya bagi klien yang terhubung melalui jaringan.
SKPL	Merupakan spesifikasi kebutuhan dari perangkat lunak yang akan dikembangkan.
UML (Unified Modeling Language)	Merupakan bahasa standart untuk melakukan visualisasi, spesifikasi, pembangunan, dan dokumentasi untuk sebuah software.
User Interface	Merupakan antarmuka yang akan menghubungkan *user* dengan sebuah sistem/perangkat lunak yang ada.
Use Case Diagram	Merupakan diagram yang digunakan untuk membantu menemukan obyek, kelas, dan relasi, serta menggambarkan kebutuhan dengan melihat bagaimana sistem dibutuhkan dan siapa penggunanya.

1.4. Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

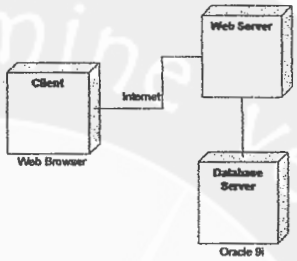
1. Alan M.Davis, Software Requirement, Prentice Hall, International Edition, 1993.
2. Bennet Simon, McRobb Steve, Farmer Ray, *Object-Oriented System Analysis and Design Using UML*, McGraw-Hill Companies, 2002.
3. Boggs Wendy, Boggs Michael, *Mastering UML with Rational Rose 2002*, SYBEX Inc, 2002.
4. Deitel, *C# How to Program*, Prentice-Hall Inc, 2002.
5. MSDN Library-October 2005, Microsoft, 2005.
6. Presman Roger S., *Rekayasa Perangkat Lunak*, McGraw-Hill Book Co., Andi Yogyakarta, 1997.

1.5. Deskripsi umum (*Overview*)

Dokumen DPPL ini terdiri dari 4 bab. Bab pertama adalah **Pendahuluan**, yang berisi deskripsi dokumen. Bab kedua adalah **Deskripsi Perancangan Arsitektural**, yang berisi deskripsi arsitektur sistem. Bab ketiga adalah **Deskripsi Perancangan Persistent Data**, yang berisi deskripsi data-data yang akan disimpan pada *persistent storage*. Bab keempat adalah **Deskripsi Perancangan Antarmuka**, yang berisi deskripsi rancangan GUI yang digunakan sistem untuk berinteraksi dengan user.

2. Deskripsi Perancangan Arsitektural

2.1. Deployment Diagram



Gambar 2.1 *Deployment Diagram CRM*

Deployment diagram ini dibuat untuk menunjukkan semua node pada sistem, hubungan di antara mereka, dan proses yang akan dijalankan di masing-masing *node*.

2.1.1 *Node : Client*

Client merupakan komputer yang digunakan oleh user untuk melakukan pengelolaan terhadap dokumen. Proses yang ada didalamnya adalah *Web browser*, digunakan untuk melakukan permintaan proses yang dijalankan pada aplikasi *CRM*.

2.1.2 *Node : Web Server*

Web Server merupakan komputer yang menyediakan layanan ke web bagi *client* yang mengakses Internet, dimana Web server ini akan mengakses basis data untuk operasi yang melibatkan data-data pada basis data.

2.1.3 *Node : Database Server*

Database Server merupakan komputer yang dipakai untuk menyediakan koneksi ke basis data dan mengautentikasi *Web server* dan tidak akan mengizinkannya melihat informasi atau menjalankan prosedur kecuali jika ia memiliki hak yang sesuai.

2.2. *Design Class*

2.2.1. Pengantar

Nama class yang digunakan dalam design class adalah nama *class* yang *valid*, termasuk nama *package*-nya. Untuk *class-class* yang berasal dari *framework* .NET juga digunakan nama *class* dengan *package* lengkap. Untuk penjelasan tipe data yang utuh dapat dilihat pada bagian deskripsi *class*, sedangkan gambar *design class* tidak akan menggunakan nama *package* yang lengkap.

Stereotype yang digunakan dalam *design class* adalah:

a. <<boundary>>

Boundary class merupakan *class* yang berfungsi untuk menghubungkan sistem dengan *user* di luar sistem.

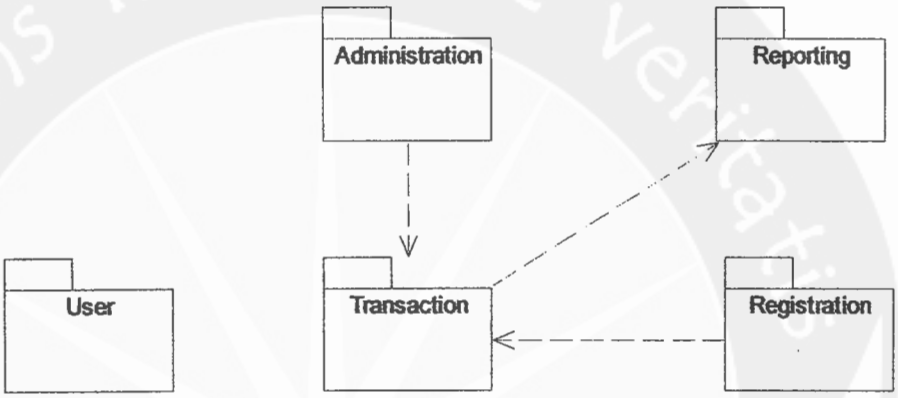
b. <<control>>

Control *class* adalah suatu *class* yang objeknya melakukan interaksi antar sekelompok objek lain. *Control class* biasanya memiliki karakteristik yang spesifik untuk satu *use case*, dan *object class* ini biasanya hanya aktif pada realisasi *use case*.

c. <<entity>>

Entity class adalah *class* yang bersifat pasif, dalam arti *class* tersebut tidak memulai interaksi dengan *class* lain. *Entity class* ini biasanya merepresentasikan suatu obyek yang disimpan dalam *persistent storage*.

2.2.2. *Package Dependency*

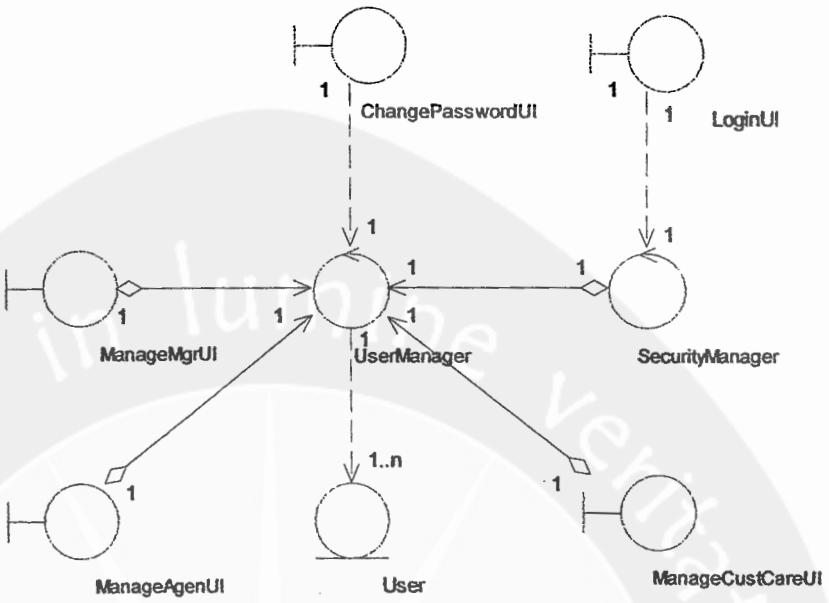


Gambar 2.2 *Package Dependency CRM*

2.2.3. *Package User*

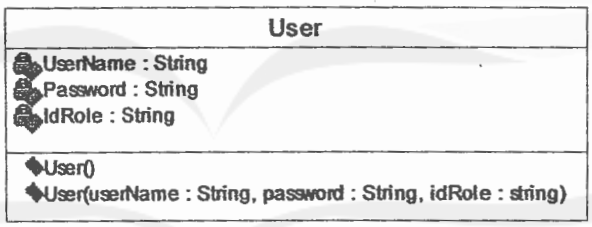
Package ini menyediakan *class-class* yang diperlukan untuk melakukan pengelolaan yang berhubungan dengan data pengguna/*user*.

2.2.3.1. *Class Diagram Package CRM.User*



Gambar 2.3 *Diagram Package CRM.User*

2.2.3.2. *Class CRM.User.User*



Gambar 2.4 *Class CRM.User.User*

Deskripsi

Class ini merepresentasikan pengguna/*user* perangkat lunak *CRM* ini.

Atribut

- *- UserName : String*

  Merepresentasikan *user name* pengguna.

- *- Password : String*

  Merepresentasikan *password* pengguna.

- *- IdRole : String*

  Merepresentasikan *id role* (hak akses) pengguna.

Method

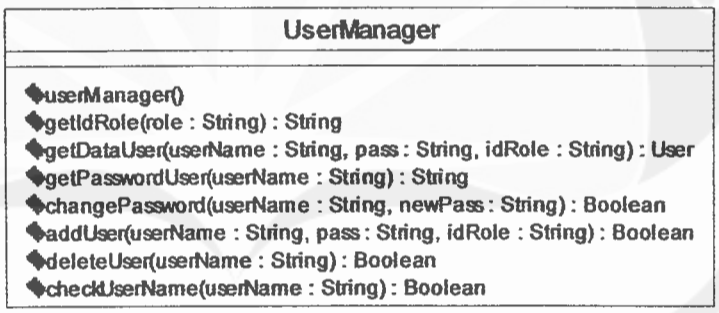
- *+ User()*

  Merupakan konstruktor class, tanpa atribut terdefinisi.

- *+ User(userName : String, password : String, idRole : String)*

  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

2.2.3.3. *Class CRM.User.UserManager*



Gambar 2.5 *Class CRM.User.UserManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan pengelolaan data *user*/pengguna pada *CRM*.

Atribut

-

Method

- + userManager()

  Merupakan konstruksi *class*, tanpa atribut yang terdefinisi.

- + getIdRole(role : *String*) : *String*

  Mendapatkan *id role* dengan memasukkan *role*.

  Parameters :

  role → *role* (hak akses) pengguna.

  Returns :

  Id role pengguna.

- + getDataUser(userName : *String*, pass : *String*, idRole : *String*) : User

  Mendapatkan atau mengecek keberadaan data *user*.

  Parameters :

  userName → user name pengguna.

  pass → password pengguna.

  id role → id role pengguna.

  Returns :

  Data pengguna meliputi user name, password, dan id role.

- + getPasswordUser(userName : *String*) : *String*

  Mendapatkan *password* dari *user name* pengguna.

  Parameters :

  userName → user name pengguna.

  Returns :

  Password pengguna.

- + changePassword(userName : *String*, newPass : *String*) : *Boolean*

  Mengganti *password user* dalam database.

  Parameters :

  userName → *user name* pengguna.

  newPass → *password* baru yang diinginkan pengguna.

  Returns :

  True jika perubahan *password* berhasil dilakukan.

- + addUser(userName : *String*, pass : *String*, idRole : *String*) : *Boolean*

  Menambah *user* baru ke dalam *database*.

  Parameters :

  userName → *user name* pengguna.

  pass → *password* pengguna.

  idRole → *id role* pengguna.

  Returns :

  True jika penambahan *user* berhasil dilakukan.

- + deleteUser(userName : *String*) : *Boolean*

  Menghapus data *user* dalam *database*.

  Parameters :

  userName → *user name* pengguna.

  Returns :

  True jika penghapusan *user* berhasil dilakukan.

- + checkUserName(userName : *String*) : *Boolean*

  Mengecek apakah *user name* ada dalam *database* atau tidak.

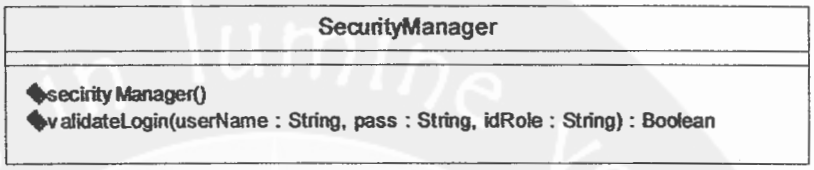
Parameters :

userName → *user name* pengguna.

Returns :

True jika penghapusan *user* berhasil dilakukan.

2.2.3.4. *Class CRM.User.SecurityManager*



Gambar 2.6 *Class CRM.User.SecurityManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan proses *login* oleh *user*/pengguna pada *CRM*.

Atribut

-

Method

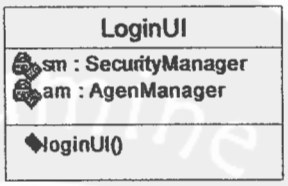
- + securityManager()
  Merupakan konstruksi *class*, tanpa atribut yang terdefinisi.
- + validateLogin(userName : *String*, pass : *String*, idRole : *String*) : *Boolean*
  Mengecek validitas *login* yang dilakukan pengguna.
  Parameters :
  userName → *user name* pengguna.
  pass → *password* pengguna.

idRole → *id role* pengguna.

Returns :

True jika data *login* yang dimasukkan adalah *valid.*

2.2.3.5. *Class CRM.User.LoginUI*



Gambar 2.7 *Class CRM.User.LoginUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses *login* pengguna.

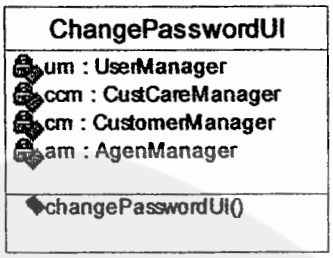
Atribut

- - sm : SecurityManager
  Merepresentasikan *object control class* untuk pengelolaan keamanan saat *login.*
- - am : AgenManager
  Merepresentasikan *object control class* untuk pengelolaan agen saat *login.*

Method

- + loginUI
  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.2.3.6. *Class CRM.User.ChangePasswordUI*



Gambar 2.8 *Class CRM.User.ChangePasswordUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses perubahan password pengguna.

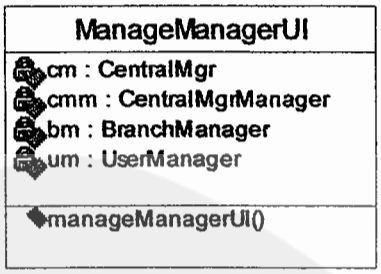
Atribut

- - um : UserManager
  Merepresentasikan *object control class* untuk pengelolaan *user* saat perubahan password.
- - ccm : CustCareManager
  Merepresentasikan *object control class* untuk pengelolaan *customer care* saat perubahan password.
- - cm : CustomerManager
  Merepresentasikan *object control class* untuk pengelolaan *customer* saat perubahan password.
- - am : AgenManager
  Merepresentasikan *object control class* untuk pengelolaan agen saat perubahan password.

Method

- + changePasswordUI()
  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.2.3.7. *Class CRM.User.ManageCentralMgrUI*



Gambar 2.9 *Class CRM.User.ManageCentralMgrUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *central manager*. Hubungannya dengan pengelolaan *user* adalah ketika dilakukan add dan delete pada *central manager*, data *user* juga akan ikut terpengaruh karena *central manager* juga termasuk dalam *user*.

Atribut

- - cm : CentralMgr
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *central manager*.
- - ccm : CentralMgrManager
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *central manager*.
- - bm : BranchManager
  Merepresentasikan *object control class* untuk pengelolaan data kantor cabang saat melakukan perubahan terhadap data *central manager*.

- - um : UserManager

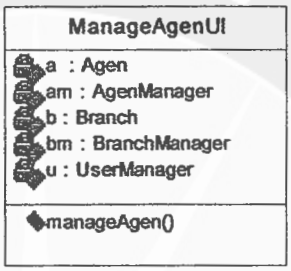
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data central *manager*.

Method

- + manageManagerUI()

  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.2.3.8. *Class CRM.User.ManageAgenUI*



Gambar 2.10 *Class CRM.User.ManageAgenUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan agen. Hubungannya dengan pengelolaan *user* adalah ketika dilakukan *add* dan *delete* pada agen, data *user* juga akan ikut terpengaruh karena agen juga termasuk dalam *user*.

Atribut

- - a : Agen

  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data agen.

- - am : AgenManager

  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data agen.

- - b : Branch

  Merepresentasikan *object control class* untuk pengelolaan data kantor cabang saat melakukan perubahan terhadap data agen.

- - bm : BranchManager

  Merepresentasikan *object control class* untuk pengelolaan data kantor cabang saat melakukan perubahan terhadap data agen.

- - um : UserManager

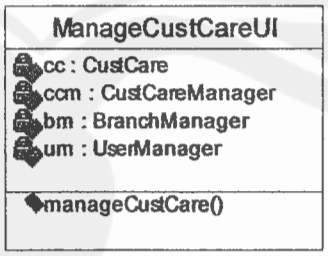
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data agen.

Method

- + manageAgenUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.3.9. *Class CRM.User.ManageCustCareUI*



Gambar 2.11 *Class CRM.User.ManageCustCareUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *customer care*. Hubungannya dengan pengelolaan *user* adalah ketika dilakukan *add* dan *delete* pada agen, data *user* juga akan ikut terpengaruh karena *customer care* juga termasuk dalam *user*.

Atribut

- - cc : CustCare
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *customer care*.
- - ccm : CustCareManager
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *customer care*.
- - bm : BranchManager
  Merepresentasikan *object control class* untuk pengelolaan data kantor cabang saat melakukan perubahan terhadap data *customer care*.
- - um : UserManager
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *customer care*.

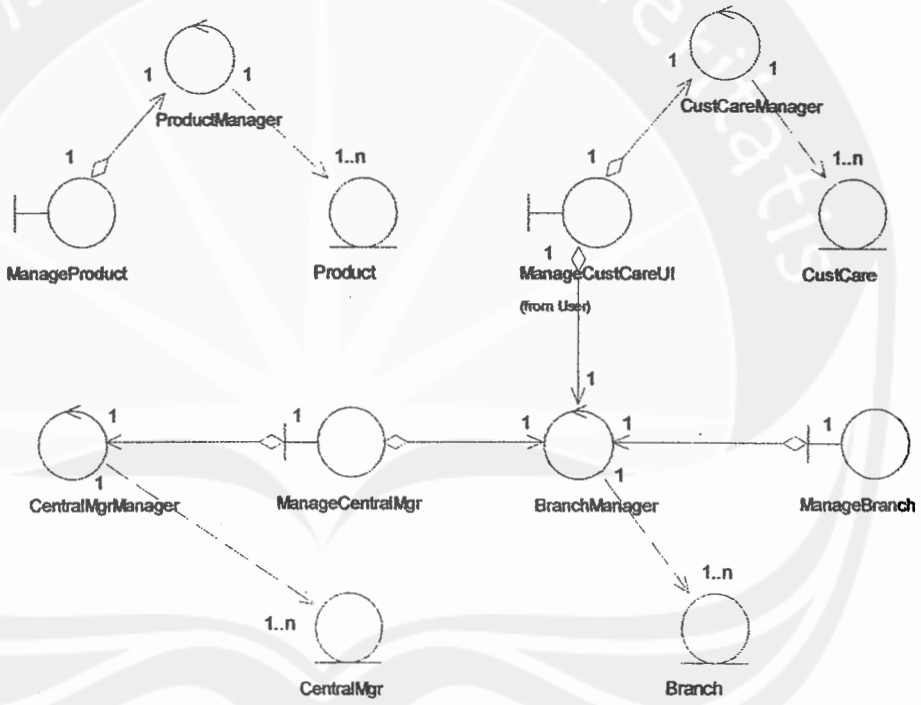
Method

- + manageCustCareUI()
  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.2.4. *Package Administration*

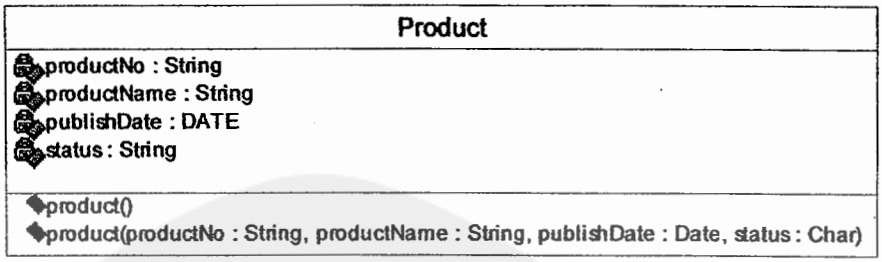
Package ini menyediakan *class-class* yang diperlukan untuk melakukan pengelolaan yang berhubungan dengan kegiatan administrasi perangkat lunak, yaitu *manage customer care, manage central manager, manage branch,* dan *manage product.*

2.2.4.1. *Class Diagram Package CRM.Administration*



Gambar 2.12 *class diagram package CRM.Administration*

2.2.4.2. *Class CRM.Administration.Product*



Gambar 2.13 *class diagram package CRM.Administration.Product*

Deskripsi

Class ini merepresentasikan product di dalam *CRM*.

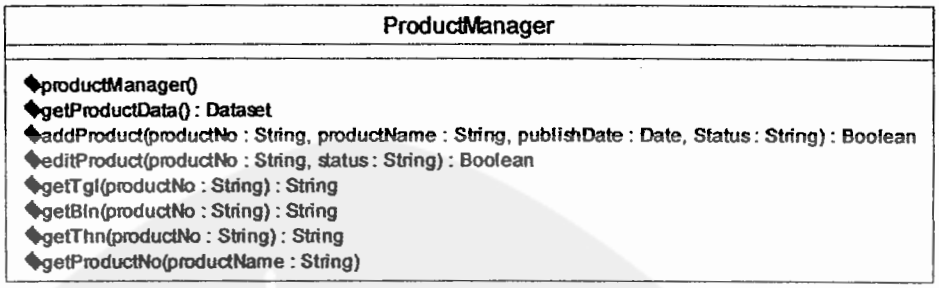
Atribut

- - productNo : *String*
  Merepresentasikan nomor produk asuransi.
- - productName : *String*
  Merepresentasikan nama produk asuransi.
- - publishDate : *Date*
  Merepresentasikan tanggal saat produk yang bersangkutan di-*publish*.
- - status : *String*
  Merepresentasikan status dari produk tersebut, aktif atau tidak aktif.

Method

- + product()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.
- + product(productNo : *String*, productName : *String*, publishDate : *Date*, status : *String*)
  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

2.2.4.3. *Class CRM.Administration.ProductManagement*



Gambar 2.14 *class diagram package CRM.Administration.ProductManagement*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan proses *product management* pada *CRM.*

Atribut

-

Method

- + productManager()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + getProductData()

  Mendapatkan data produk asuransi.

  Returns :

  Sebuah *dataset*.

- + addProduct (productNo : *String*, productName : String, publishDate : *Date*, Status : *String*) : *Boolean*

  Menambahkan data produk asuransi baru.

Parameters :

productNo → nomor produk asuransi.

productName → nama produk asuransi.

publishDate → tanggal produk di-*publish*.

Status → status produk.

Returns :

True jika penambahan berhasil dilakukan.

- + editProduct (productNo : *String*, status : *String*) : *Boolean*

  Mengedit data status produk menjadi aktif atau tidak aktif.

  Parameters :

  productNo → nomor produk asuransi.

  Status → status produk

  Returns :

  True jika pengeditan data berhasil dilakukan.

- + getTgl ()

  Mendapatkan tanggal *publish date*.

  Returns :

  String tanggal.

- + getBln()

  Mendapatkan bulan *publish date*.

  Returns :

  String bulan.

- + getThn()

  Mendapatkan tahun *publish date*.

  Returns :

  String tahun.

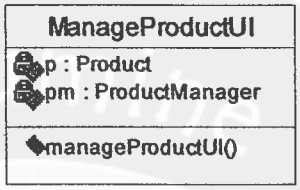
- + getProductNo()

  Mendapatkan nomor produk dari nama produk.

  Returns :

  String nomor produk.

2.2.4.4. *Class CRM.Administration.ManageProductUI*



Gambar 2.15 *class diagram package CRM.Administration.ManageProductUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *branch*/kantor cabang.

Atribut

- - b : Product

  Merepresentasikan *object control class* untuk pengelolaan *product* saat melakukan perubahan terhadap data *product*.

- - bb : ProductManager

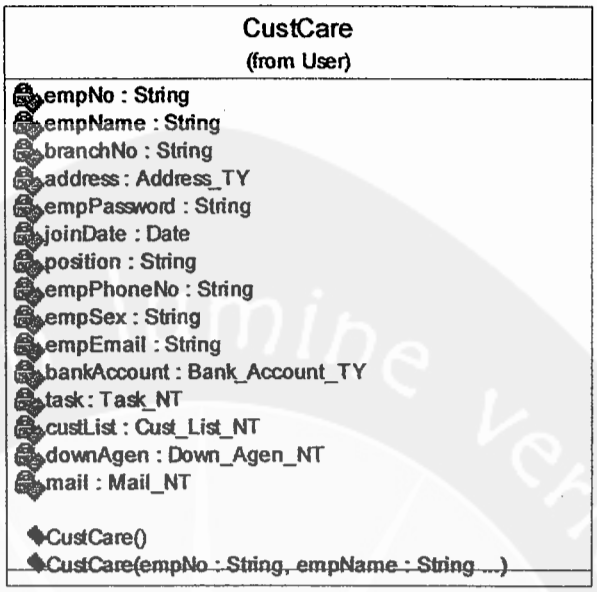
  Merepresentasikan *object control class* untuk pengelolaan *product* saat melakukan perubahan terhadap data *product*.

Method

- + manageProductUI()

  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.2.4.5. *Class CRM.Administration.CustCare*



Gambar 2.16 *class diagram package CRM.Administration.CustCare*

Deskripsi

Class ini merepresentasikan *Customer Care* di dalam *CRM*.

Atribut

- - empNo : *String*
  Merepresentasikan nomor *customer care*.
- - empName : *String*
  Merepresentasikan nama *customer care*.
- - branchNo : *String*
  Merepresentasikan nomor kantor cabang tempat *customer care* bekerja.
- - address : Address_TY
  Merepresentasikan alamat (sebuah tipe) dari *customer care*.

- - empPassword : *String*

  Merepresentasikan password yang nantinya digunakan untuk *login* sebagai *customer care*.

- - joinDate : *Date*

  Merepreentasikan tanggal dimana *customer care* yang bersangkutan bergabung dengan perusahaan.

- - position : *String*

  Merepresentasikan posisi dalam perusahaan.

- - empPhoneNo : *String*

  Merepresentasikan nomor telepon *customer care*.

- - empSex : *String*

  Merepresentasikan jenis kelamin *customer care*.

- - empEmail : *String*

  Merepresentasikan alamat email *customer care*.

- - bankAccout : *String*

  Merepresentasikan data rekening bank yang dimiliki *customer care*.

- - task : Task_NT

  Merepresentasikan daftar tugas atau rencana harian *customer care*.

- - custList : Cust_List_NT

  Merepresentasikan daftar *customer* yang dimiliki oleh *customer care*.

- - downAgen : Down_Agen_NT

  Merepresentasikan daftar agen yang menjadi bawahan *customer care*.

- - mail : *String*

  Merepresentasikan daftar mail yang masuk ke *account customer care*.

Method

- + CustCare()

  Merupakan konstruktor *class*.

- + CustCare(empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, empPassword : *String*, joinDate : *Date*, position : *String*, empSex : *String*, empEmail : *String*, bankAccount : Bank_Account_TY, task : Task_TY, custList : Cust_List_NT, downAgen : Down_Agen_NT, mail : Mail_NT)

  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

  Parameters :

  empNo → nomor pegawai *customer care*.

  empName → nama *customer care*.

  branchNo → nomor kantor cabang.

  address → alamat *customer care*.

  empPassword → password *customer care*.

  joinDate → tanggal menjadi pegawai.

  position → posisi dalam perusahaan.

  empSex → jenis kelamin *customer care*.

  empEmail → alamat email *customer care*.

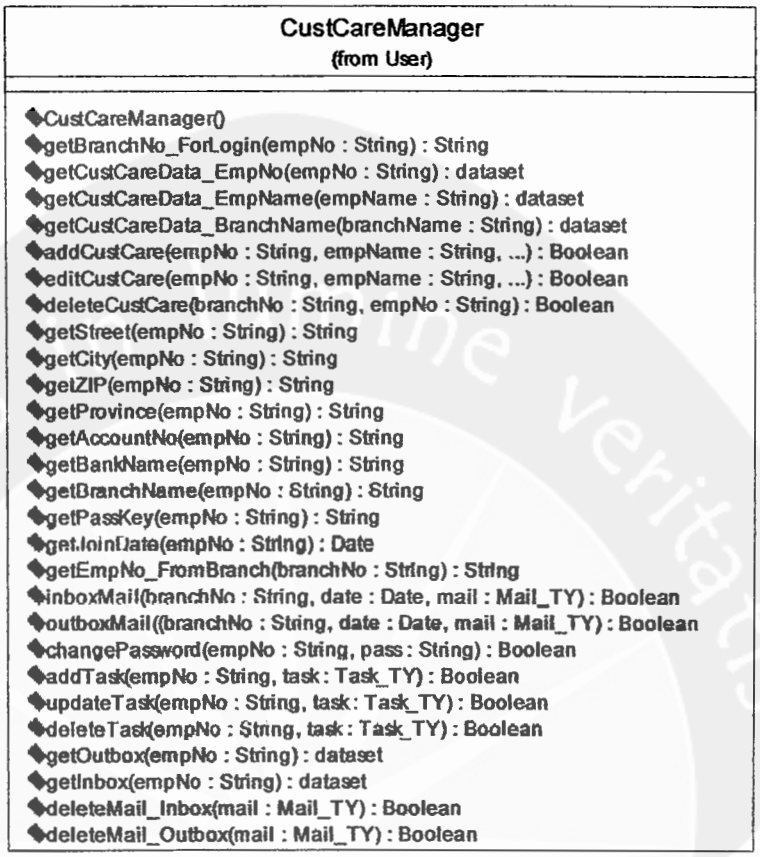
  bankAccount → rekening bank *customer care*.

  task → daftar tugas/jadwal *customer care*.

  custList → daftar *customer*.

  mail → daftar *mail* di *account customer care*.

2.2.4.6. *Class CRM.Administration.CustCareManager*



Gambar 2.17 *class diagram package CRM.Administration.CustCareManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan proses *customer care management* pada *CRM.*

Atribut

-

: Bank_Account_TY, passKey : *String*) : *Boolean*

Melakukan pengeditan data *customer care*.

Parameters :

empNo → nomor pegawai *customer care*.

empName → nama *customer care*.

branchNo → nomor kantor cabang.

address → alamat *customer care*.

joinDate → tanggal menjadi pegawai.

phoneNo → nomor telepon *customer care*.

email → alamat email *customer care*.

bankAccount → rekening bank *customer care*.

passKey → *password* untuk melakukan login sebagai *customer care*.

Returns :

True jika data berhasil diedit.

- + deleteCustCare(branchNo : *String*, empNo : *String*) : *Boolean*

  Paramaters :

  branchNo → nomor kantor cabang tempat *customer care* bekerja.

  empNo → nomor pegawai *customer care*.

  Returns :

  True jika data berhasil dihapus.

- + getStreet(empNo : *String*) : *String*

  Mendapatkan nama jalan.

  Paramaters :

  empNo → nomor pegawai *customer care*.

Returns :

String jalan (bagian dari tipe alamat).

- + getCity(empNo : *String*) : *String*

  Mendapatkan nama kota.

  Paramaters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String kota (bagian dari tipe alamat).

- + getZIP(empNo : *String*) : *String*

  Mendapatkan kodepos.

  Paramaters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String kodepos (bagian dari tipe alamat).

- + getProvince(empNo : *String*) : *String*

  Mendapatkan nama propinsi.

  Paramaters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String propinsi (bagian dari tipe alamat).

- + getAccountNo(empNo : *String*) : *String*

  Mendapatkan nomor rekening.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String nomor rekening.

- + getBankName(empNo : *String*) : *String*

  Mendapatkan nama bank.

Parameters :

empNo → nomor pegawai *customer care*.

Returns :

String nama bank.

- + getBranchName(empNo : *String*) : *String*

  Mendapatkan nama cabang bank.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String nama cabang bank.

- + getPassKey(empNo : *String*) : *String*

  Mendapatkan password *customer care*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String password.

- + getJoinDate(empNo : *String*) : *String*

  Mendapatkan tanggal *customer care* menjadi pegawai.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  String tanggal menjadi pegawai.

- + getEmpNo_FromBranch(branchNo : *String*) : *String*

  Mendapatkan nomor pegawai dari kode kantor cabang.

  Parameters :

  branchNo → kode kantor cabang.

Returns :

String nomor pegawai.

- + inboxMail(branchNo : *String*, date : *Date*, mail : Mail_TY) : *Boolean*

  Menambahkan data *inbox mail* ke *database*.

  Paramaters :

  branchNo → nomor kantor cabang,

  date → tanggal *inbox mail* masuk.

  mail → data mail yang masuk.

  Returns :

  True jika mail berhasil disimpan.

- + outboxMail(branchNo : *String*, date : *Date*, mail : Mail_TY) : *Boolean*

  Menambahkan data *outbox mail* ke *database*.

  Paramaters :

  branchNo → nomor kantor cabang,

  date → tanggal *inbox mail* masuk.

  mail → data mail yang masuk.

  Returns :

  True jika mail berhasil terkirim.

- + changePassword(empNo : String, pass : String) : *Boolean*

  Melakukan pengeditan password.

  Parameters :

  empNo → nomor pegawai *customer care*.

  pass → password baru.

  Returns :

  True jika password berhasil diedit.

- + addTask(empNo : *String*, task : Task_TY) : *Boolean*

  Menambahkan data *task*/jadwal *customer care* ke *database*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  task → jadwal *customer care*.

  Returns :

  True jika data task berhasil ditambahkan.

- + updateTask(empNo : *String*, task : Task_TY) : *Boolean*

  Mengedit data *task*/jadwal *customer care* ke *database*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  task → jadwal *customer care*.

  Returns :

  True jika data task berhasil di-*update*.

- + deleteTask(empNo : *String*, task : Task_TY) : *Boolean*

  Menghapus data *task*/jadwal *customer care* ke *database*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  task → jadwal *customer care*.

  Returns :

  True jika data task berhasil dihapus.

- + getOutbox(empNo : String) : Dataset

  Mendapatkan data *outbox mail*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  Dataset outbox mail.

- + getInbox(empNo : String) : Dataset

  Mendapatkan data *inbox mail*.

  Parameters :

  empNo → nomor pegawai *customer care*.

  Returns :

  Dataset inbox mail.

- + deleteMail_Inbox(mail : Mail_TY) : *Boolean*

  Menghapus *inbox mail*.

  Parameters :

  mail → data mail yang ingin dihapus.

  Returns :

  True jika data berhasil dihapus.

- + deleteMail_Outbox(mail : Mail_TY) : *Boolean*

  Menghapus *outbox mail*.

  Parameters :

  mail → data mail yang ingin dihapus.

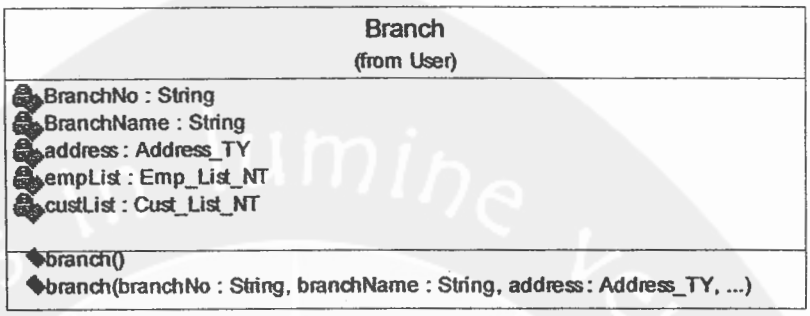
  Returns :

  True jika data berhasil dihapus.

2.2.4.7. *Class CRM.Administration.ManageCustCareUI*

Lihat : *Class CRM.User.ManageCustCareUI*

2.2.4.8. *Class CRM.Administration.Branch*



Gambar 2.18 *class diagram package CRM.Administration.Branch*

Deskripsi

Class ini merepresentasikan *Branch* di dalam *CRM*.

Atribut

- - branchNo : *String*
  Merepresentasikan nomor kantor cabang tempat *customer care* bekerja.
- - branchName : *String*
  Merepresentasikan nama kantor cabang tempat *customer care* bekerja.
- - address : Address_TY
  Merepresentasikan alamat (sebuah tipe) dari *customer care*.
- - empList : Emp_List_NT
  Merepresentasikan daftar agen yang posisinya berada di bawah *customer care* bersangkutan.

- - custList : Cust_List_NT

  Merepresentasikan daftar *customer* yang ada di kantor tempat *customer care* bekerja.

Method

- + Branch()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + Branch(branchNo : String, branchName : String, address : Address_TY, empList : Emp_List_NT, custList : Cust_List_NT)

  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

  Parameters :

  branchNo → nomor kantor cabang.

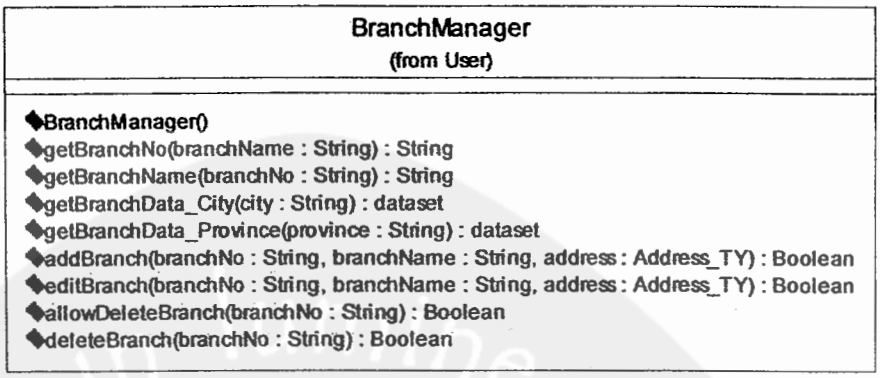
  branchName → nama kantor cabang.

  address → alamat *customer care*.

  empList → daftar agen yang posisinya berada di bawah *customer care* bersangkutan.

  custList → daftar *customer* di cabang yang bersangkutan.

2.2.4.9. *Class CRM.Administration.BranchManager*



Gambar 2.19 *class diagram package CRM.Administration.BranchManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan proses *customer care management* pada *CRM.*

Atribut

-

Method

- + BranchManager()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.
- + getBranchNo (branchName : *String*) : *String*
  Mendapatkan nomor kantor cabang.
  Parameters :
  branchName → nama kantor cabang.
  Returns :
  String nomor kantor cabang.
- + getBranchName (branchNo : *String*) : *String*
  Mendapatkan nama kantor cabang.

Parameters :

branchNo → nomor kantor cabang.

Returns :

String nama kantor cabang.

- + getBranchData_City (city : *String*) : Dataset.

  Mendapatkan data kantor cabang dari kota lokasinya.

  Parameters :

  city → kota lokasi kantor cabang berada.

  Returns :

  Sebuah dataset yang berisi data kantor cabang.

- + getBranchData_Province (province : String) : Dataset

  Mendapatkan data kantor cabang dari propinsi lokasinya

  Parameters :

  propinsi → propinsi lokasi kantor cabang berada.

  Returns :

  Sebuah *dataset* yang berisi data kantor cabang.

- + addBranch (branchNo : *String*, branchName : *String*, address : Address, empList : Emp_List_NT, custList : Cust_List_NT) : *Boolean*

  Melakukan penambahan kantor cabang.

Parameters :

branchNo → nomor kantor cabang.

branchName → nama kantor cabang.

address → alamat kantor cabang.

empList → daftar pegawai (agen dan *customer care*) pada sebuah kantor cabang.

custList → daftar *customer* pada sebuah kantor cabang.

Returns :

True jika data berhasil ditambahkan.

- + editBranch (branchNo : *String*, branchName : *String*, address : Address, empList : Emp_List_NT, custList : Cust_List_NT) : *Boolean*

  Melakukan pengeditan data kantor cabang.

  Parameters :

  branchNo → nomor kantor cabang.

  branchName → nama kantor cabang.

  address → alamat kantor cabang.

  empList → daftar pegawai (agen dan *customer care*) pada sebuah kantor cabang.

  custList → daftar *customer* pada sebuah kantor cabang.

  Returns :

  True jika data berhasil diedit.

- + allowDeleteBranch (branchNo : *String*) : *Boolean*

  Mengetahui apakah kantor cabang masih memiliki pegawai dan customer atau tidak.

Parameters :

branchNo → nomor kantor cabang.

Returns :

True jika masih ada pegawai atau customer di cabang yang bersangkutan.

- + deleteBranch (branchNo : String) : Boolean

  Melakukan penghapusan terhadap kantor cabang.

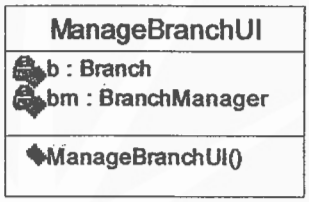
  Parameters :

  branchNo → nomor kantor cabang.

  Returns :

  True jika data berhasil dihapus.

2.2.4.10. *Class CRM.Administration.ManageBranchUI*



Gambar 2.20 *class diagram package CRM.Administration.ManageBranchUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *branch* atau kantor cabang.

Atribut

- - b : Branch

  Merepresentasikan *object control class* untuk pengelolaan *branch* saat melakukan perubahan terhadap data *branch*.

- - bm : BranchManager

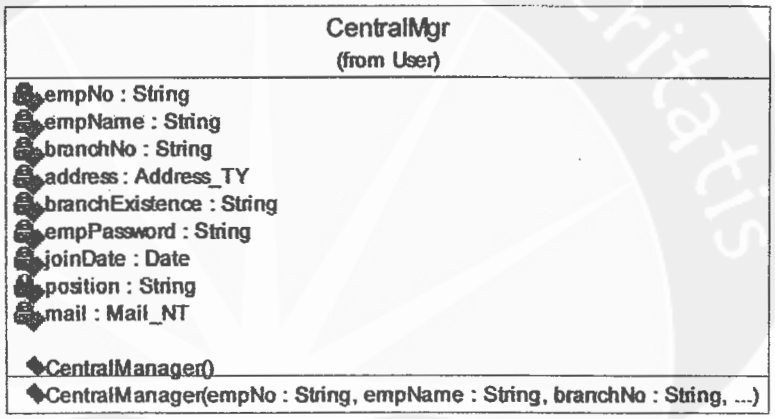
  Merepresentasikan *object control class* untuk pengelolaan data *branch* saat melakukan perubahan terhadap data *branch*.

Method

- + manageBranchUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.4.11. *Class CRM.Administration.CentralMgr*



Gambar 2.21 *class diagram package CRM.Administration.CentralMgr*

Deskripsi

Class ini merepresentasikan *Cental Manager* di dalam *CRM*.

Atribut

- - empNo : *String*

  Merepresentasikan nomor pegawai yang dimiliki manajer pusat.

- - empName : *String*

  Merepresentasikan nama pegawai yang dimiliki manajer pusat.
- - branchNo : *String*

  Merepresentasikan nomor kantor cabang tempat manajer pusat bekerja.
- - address : Address_TY

  Merepresentasikan alamat manajer pusat.
- - branchExistence : *String*

  Merepresentasikan keberadaan manajer pusat.
- - empPassword : *String*

  Merepresentasikan *password* untuk login sebagai manajer pusat.
- - joinDate : *Date*

  Merepresentasikan tanggal manajer pusat menjadi pegawai.
- - position : *String*

  Merepresentasikan posisi pegawai.
- - mail : Mail_NT

  Merepresentasikan daftar *mail* yang dimiliki manajer pusat dalam database.

Method

- + CentralManager()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.
- + CentralManager (empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, branchExistence : *String*,

empPassword : *String*, joinDate : Date, position : *String*, mail : Mail_NT)

Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

Parameters :

empNo → nomor pegawai.

empName → nama pegawai

branchNo → nomor kantor cabang

address → alamat manajer pusat

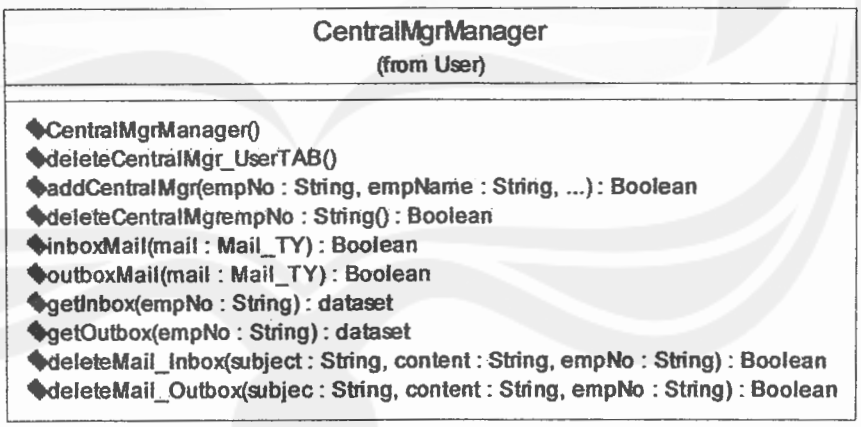
branchExistence → keberadaan manajer pusat

empPassword → password untuk login sebagai manajer pusat

joinDate → tanggal menjadi pegawai

mail → daftar mail yang dimiliki manajer pusat.

2.2.4.12. *Class CRM.Administration.CentralMgrManager*



Gambar 2.22 *class diagram package CRM.Administration.CentralMgrManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan proses *central manager management* pada *CRM.*

Atribut

-

Method

- + CentralMgrManager()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.
- + deleteCentralMgr_UserTAB ()
  Melakukan penghapusan data central manager pada tabel *user*.
- + addCentralMgr (empNo : *String*, empName : String, branchNo : String, address : Address_TY, branchExistence : String, empPassword : String, joinDate : Date, position : String, mail : Mail_NT) : *Boolean*
  Parameters :
  empNo → nomor pegawai manajer pusat.
  empName → nama manajer pusat.
  address → alamat manajer pusat.
  branchExistence → kantor tempat manajer pusat berada.
  empPassword → password yang digunakan untuk *login* sebagai manajer pusat.
  joinDate → tanggal manajer pusat menjadi pegawai.
  position → posisi dalam perusahaan.

mail → daftar *mail* yang dimiliki oleh manajer pusat.

Returns :

True jika penambahan data berhasil dilakukan.

- + deleteCentralMgr (empNo : *String*) : *Boolean*

  Melakukan penghapusan terhadap data manajer cabang.

  Parameters :

  empNo → nomor pegawai manajer pusat.

  Returns :

  True jika pengeditan data berhasil dilakukan.

- + inboxMail (mail : Mail_NT) : *Boolean*

  Melakukan penambahan data *mail* dengan status *inbox*.

  Parameters :

  Mail → data *mail inbox*.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + outboxMail (mail : Mail_NT) : *Boolean*

  Melakukan penambahan data *mail* dengan status *outbox*.

  Parameters :

  Mail → data mail *outbox*.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + getInbox (empNo : *String*) : *Dataset*

  Mendapatkan data *mail* dengan status *inbox*.

- + getOutbox (empNo : *String*) : *Dataset*

  Mendapatkan data *mail* dengan status *outbox*.

- + deleteMail_Inbox (subject : *String*, content : String, empNo : *String*) : *Boolean*

  Melakukan penghapusan terhadap data mail dengan status *inbox*.

  Parameters :

  subject → subject mail.

  content → isi mail.

  empNo → nomor pegawai.

  Returns :

  True jika *inbox mail* berhasil dihapus.

- + deleteMail_Outbox (subject : *String*, content : String, empNo : *String*) : *Boolean*

  Melakukan penghapusan terhadap data mail dengan status *outbox*.

  Parameters :

  subject → subject mail.

  content → isi mail.

  empNo → nomor pegawai.

  Returns :

  True jika *outbox mail* berhasil dihapus.

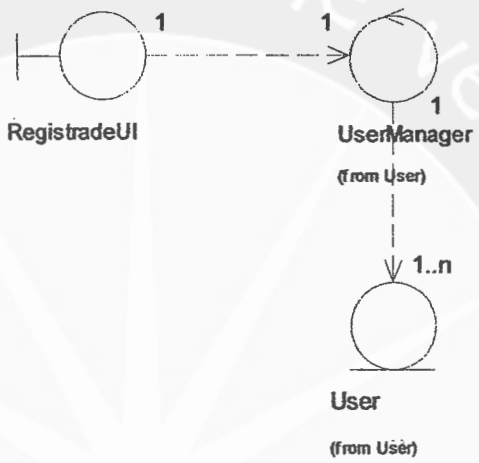
2.2.4.13. *Class CRM.Administration.ManageCentralMgrUI*

Lihat : *Class CRM.User.ManageCentralMgrUI*

2.2.5. *Package Registrade*

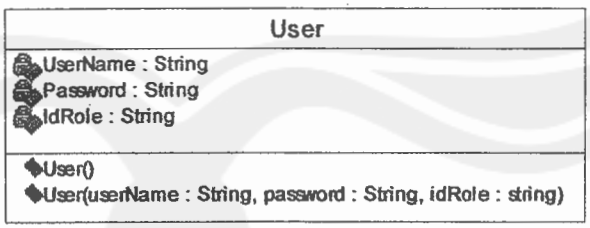
Package ini menyediakan *class-class* yang diperlukan untuk melakukan pengelolaan yang berhubungan dengan kegiatan transaksi perangkat lunak, yaitu *manage agen, manage customer, manage task*, dan *manage mail*, dan ubah *password.*

2.2.5.1. *Class Diagram Package CRM.Registrade*



Gambar 2.23 *class diagram package CRM.Registrade*

2.2.5.2. *Class CRM.Registrade.User*



Gambar 2.24 *Class CRM.Registrade.User*

Deskripsi

Class ini merepresentasikan pengguna/*user* perangkat lunak *CRM* ini.

Atribut

- *- UserName : String*

  Merepresentasikan *user name* pengguna.

- *- Password : String*

  Merepresentasikan *password* pengguna.

- *- IdRole : String*

  Merepresentasikan *id role* (hak akses) pengguna.

Method

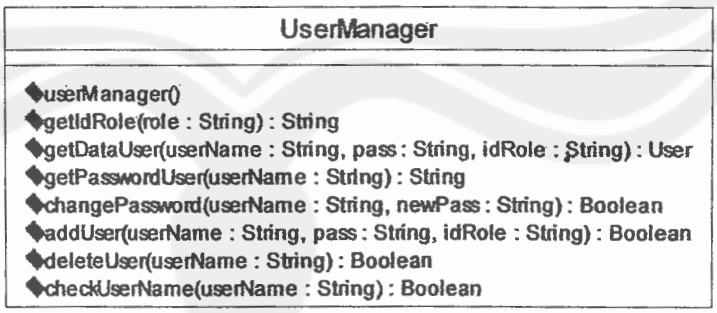
- *+ User()*

  Merupakan konstruktor class, tanpa atribut terdefinisi.

- *+ User(userName : String, password : String, idRole : String)*

  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

2.2.5.3. *Class CRM.Registrade.UserManager*



Gambar 2.25 *Class CRM.Registrade.UserManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan pengelolaan data *user*/pengguna pada *CRM*.

Atribut

-

Method

- + userManager()
  Merupakan konstruksi *class*, tanpa atribut yang terdefinisi.
- + getIdRole(role : *String*) : *String*
  Mendapatkan *id role* dengan memasukkan *role*.
  Parameters :
  role → *role* (hak akses) pengguna.
  Returns :
  Id role pengguna.
- + getDataUser(userName : *String*, pass : *String*, idRole : *String*) : User
  Mendapatkan atau mengecek keberadaan data *user*.
  Parameters :
  userName → user name pengguna.
  pass → password pengguna.
  id role → id role pengguna.
  Returns :
  Data pengguna meliputi user name, password, dan id role.
- + getPasswordUser(userName : *String*) : *String*
  Mendapatkan *password* dari *user name* pengguna.

Parameters :

userName → user name pengguna.

Returns :

Password pengguna.

- + changePassword(userName : *String*, newPass : *String*) : *Boolean*

  Mengganti *password user* dalam database.

  Parameters :

  userName → *user name* pengguna.

  newPass → *password* baru yang diinginkan pengguna.

  Returns :

  True jika perubahan *password* berhasil dilakukan.

- + addUser(userName : *String*, pass : *String*, idRole : *String*) : *Boolean*

  Menambah *user* baru ke dalam *database*.

  Parameters :

  userName → *user name* pengguna.

  pass → *password* pengguna.

  idRole → *id role* pengguna.

  Returns :

  True jika penambahan *user* berhasil dilakukan.

- + deleteUser(userName : *String*) : *Boolean*

  Menghapus data *user* dalam *database*.

  Parameters :

  userName → *user name* pengguna.

Returns :

True jika penghapusan *user* berhasil dilakukan.

- + checkUserName(userName : *String*) : *Boolean*

  Mengecek apakah *user name* ada dalam *database* atau tidak.

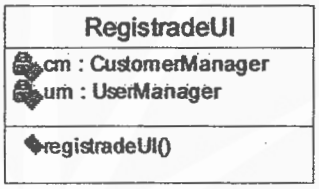
  Parameters :

  userName → *user name* pengguna.

  Returns :

  True jika penghapusan *user* berhasil dilakukan.

2.2.5.4. *Class CRM.Registrade.RegistradeUI*



Gambar 2.26 *Class CRM.Registrade.RegistradedUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses perubahan registrasi pengguna (khususnya bagi *customer*).

Atribut

- - cm : CustomerManager

  Merepresentasikan *object control class* untuk pengelolaan *customer* saat melakukan registrasi.

- \- um : UserManager

  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan registrasi.

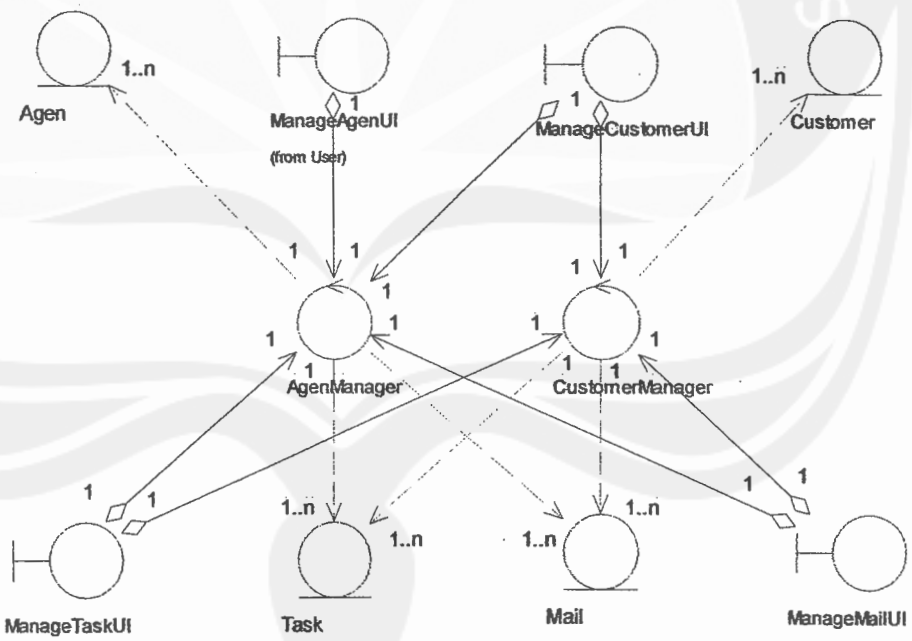
Method

- \+ registradeUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.6. *Package Transaction*

Package ini menyediakan *class-class* yang diperlukan untuk melakukan pengelolaan yang berhubungan dengan kegiatan transaksi perangkat lunak, yaitu *manage agen*, *manage customer*, *manage task*, dan *manage mail*, dan ubah *password*.



Gambar 2.27 *class diagram package CRM.Transaction*

2.2.6.1. *Class CRM.Transaction.Agen*

Gambar 2.28 *Class CRM.Transaction.Agen*

Deskripsi

Class ini merepresentasikan agen dalam perangkat lunak *CRM* ini.

Atribut

- - empNo : *String*

  Merepresentasikan nomor pegawai yang dimiliki agen.

- - empName : *String*

  Merepresentasikan nama agen.
- - branchNo : *String*

  Merepresentasikan nomor kantor cabang tempat agen bekerja.
- - address : Address_TY

  Merepresentasikan alamat agen.
- - empPassword : *String*

  Merepresentasikan *password* untuk *login* sebagai agen.
- - joinDate : *Date*

  Merepresentasikan tanggal agen menjadi pegawai.
- - position : *String*

  Merepresentasikan posisi pegawai.
- - empPhoneNo : *String*

  Merepresentasikan nomor telepon agen.
- - empBirthPlace : *String*

  Merepresentasikan tempat lahir agen.
- - empBirthDate : *Date*

  Merepresentasikan tanggal lahir agen.
- - empSex : *String*

  Merepresentasikan jenis kelamin agen.
- - empEmail : *String*

  Merepresentasikan alamat email agen.
- - empMarriageStatus : *String*

  Merepresentasikan status pernikahan agen.
- - memberState : *String*

Merepresentasikan warga negara.

- - empIdentityCardNo : *String*

  Merepresentasikan nomor kartu identitas.

- - custCareNo : *String*

  Merepresentasikan nomor pegawai *customer care* (atasan agen).

- - bankAccount : Bank_Account_TY

  Merepresentasikan data rekening bank.

- - workExp : Work_Exp_TY

  Merepresentasikan pengalaman bekerja agen.

- - education : Education_TY

  Merepresentasikan pendidikan terakhir yang diterima agen.

- - task : Task_NT

  Merepresentasikan daftar tugas/jadwal agen.

- - custList : Cust_List_NT

  Merepresentasikan daftar *customer* yang didapatkan agen bersangkutan.

- - agenSalary : Agen_Salary_TY

  Merepresentasikan gaji yang diterima agen.

- - mail : Mail_NT

  Merepresentasikan daftar *mail* yang dimiliki agen.

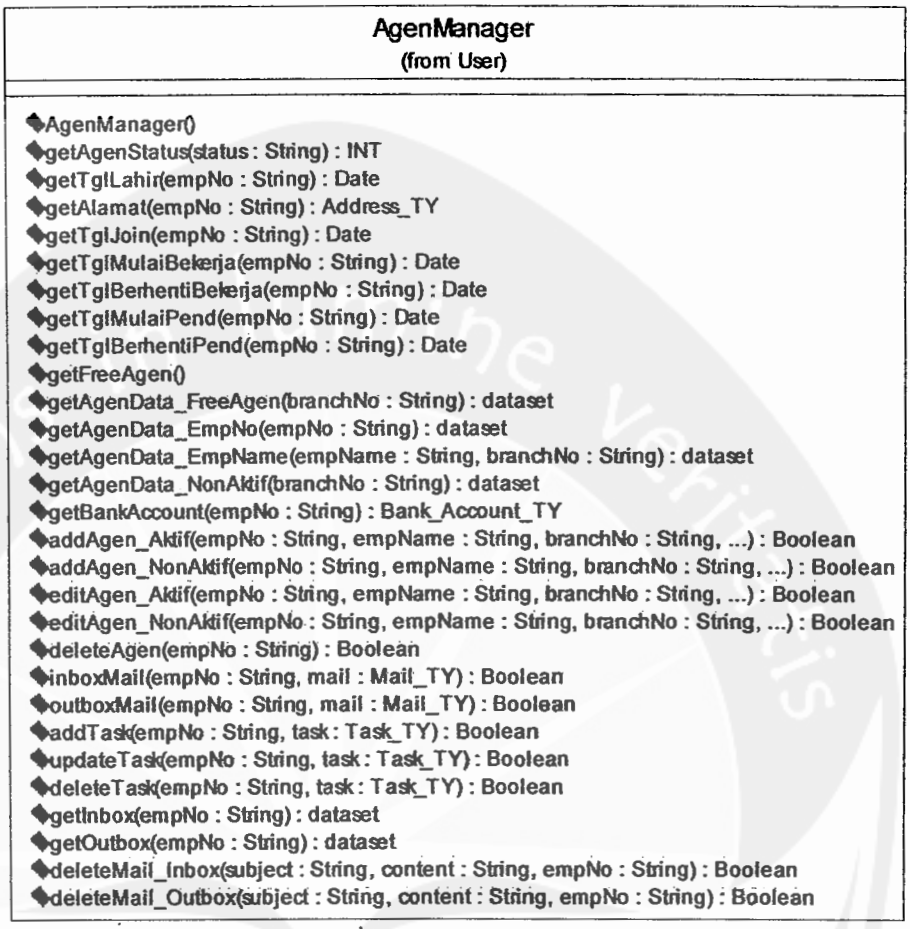
Method

- + Agen()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + Agen(empNo : *String*, empName : *String*, branchNo : *String*, empPassword : *String*, joinDate : Date, position : *String*, empPhoneNo : String, empBirthPlace : *String*, empBirthDate : *String*, empSex : *String*, empEmail : *String*, empMarriageStatus : String, empIdentityCardNo : *String*, custCareNo : *String*, bankAccount : Bank_Account_TY, workExp : Work_Exp_TY, education : Education_TY, task : Task_NT, custList : Cust_List_NT, agenSalary : Agen_Salary_NT, mail : Mail_NT)
  Merupakan konstruktor *class*, dengan atribut yang terdefinisi.

2.2.6.2. *Class CRM.Tansaction.AgenManager*



Gambar 2.29 *Class CRM.Transaction.AgenManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan pengelolaan data agen pada *CRM*.

Atribut

-

Method

- + AgenManager()

  Merupakan konstruksi *class*, tanpa atribut yang terdefinisi.

- + getAgenStatus(status : *String*) : *int*

  Mendapatkan status agen.

  Parameters :

  status → status agen, aktif atau tidak aktif.

  Returns :

  int status.

- + getTglLahir(empNo : *String*) : *Date*

  Mendapatkan tanggal lahir agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Tanggal lahir.

- + getAlamat(empNo : *String*) : Address_TY

  Mendapatkan alamat agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Alamat (tipe address).

- + getTglJoin(empNo : *String*) : *Date*

  Mendapatkan tanggal menjadi pegawai.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

Returns :

Tanggal menjadi pegawai.

- + getTglMulaiBekerja(empNo : *String*) : *Date*

  Mendapatkan tanggal mulai bekerja.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Tanggal mulai bekerja.

- + getTglBerhentiBekerja(empNo : *String*) : *Date*

  Mendapatkan tanggal berhenti bekerja.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Tanggal berhenti bekerja.

- + getTglMulaiPend(empNo : *String*) : *Date*

  Mendapatkan tanggal mulai pendidikan terakhir.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Tanggal mulai pendidikan yang terakhir.

- + getTglBerhentiPend(empNo : *String*) : *Date*

  Mendapatkan tanggal berhenti pendidikan terakhir.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

Returns :

Tanggal berhenti pendidikan yang terakhir.

- + getFreeAgen()
- + getAgenData_FreeAgen(branchNo : *String*) : *Dataset*

  Mendapatkan data agen yang tidak memiliki *customer care*.

  Parameters :

  branchNo → nomor kantor cabang.

  Returns :

  Dataset yang berisi data agen.
- + getAgenData_EmpNo(empNo : *String*) : *Dataset*

  Mendapatkan data agen berdasarkan nomor pegawai.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Dataset yang berisi data agen.
- + getAgenData_EmpName(empName : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data agen berdasarkan nama agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Dataset yang berisi data agen.
- + getAgenData_NonAktif(branchNo : *String*) : *Dataset*

  Mendapatkan data agen yang tidak aktif.

Parameters :

branchNo → nomor kantor cabang.

Returns :

Dataset yang berisi data agen.

- + getBankAccount(empNo : *String*) : Bank_Account_TY

  Mendapatkan data rekening agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Data rekening agen.

- + addAgen_Aktif(empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, empPassKey : *String*, joinDate : *Date*, position : *String*, agenStatus : *int*, phoneNo : *String*, birthPlace : *String*, birthDate : *Date*, empSex : *String*, empMail : *String*, marriageStatus : *String*, memberState : *String*, identityNo : *String*, custCareNo : *String*, bankAccount : Bank_Account_TY, workExp : Work_Exp_TY, education : Education_TY) : *Boolean*

  Melakukan penambahan data agen yang aktif.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  empName → nama agen.

  branchNo → nomor kantor cabang tempat agen bekerja.

  address → alamat agen.

empPassKey → password untuk login sebagai agen.

joinDate → tanggal menjadi pegawai.

position → posisi pegawai.

agenStatus → status agen, aktif atau tidak aktif.

phoneNo → nomor telepon agen.

birthPlace → tempat lahir agen.

birthDate → tanggal lahir agen.

empSex → jenis kelamin agen.

empMail → alamat email agen.

marriageStatus → status pernikahan agen.

memberState → kewarganegaraan agen.

identityNo → nomor kartu identitas agen.

custCareNo → nomor pegawai *customer care* atasan agen.

bankAccount → data rekening bank agen.

workExp → data pengalaman bekerja agen.

education → data pendidikan terakhir agen.

Returns :

True jika data berhasil ditambahkan.

- + addAgen_NonAktif(empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, empPassKey : String, joinDate : *Date*, position : *String*, agenStatus : *int*, phoneNo : *String*, birthPlace : *String*, birthDate : *Date*, empSex : *String*, empMail : *String*, marriageStatus : *String*, memberState : *String*, identityNo : *String*, custCareNo :

String, bankAccount : Bank_Account_TY, workExp : Work_Exp_TY, education : Education_TY) : *Boolean*

Melakukan penambahan data agen yang tidak aktif

Parameters :

empNo → nomor pegawai yang dimiliki agen.

empName → nama agen.

branchNo → nomor kantor cabang tempat agen bekerja.

address → alamat agen.

empPassKey → password untuk login sebagai agen.

joinDate → tanggal menjadi pegawai.

position → posisi pegawai.

agenStatus → status agen, aktif atau tidak aktif.

phoneNo → nomor telepon agen.

birthPlace → tempat lahir agen.

birthDate → tanggal lahir agen.

empSex → jenis kelamin agen.

empMail → alamat email agen.

marriageStatus → status pernikahan agen.

memberState → kewarganegaraan agen.

identityNo → nomor kartu identitas agen.

custCareNo → nomor pegawai *customer care* atasan agen.

bankAccount → data rekening bank agen.

workExp → data pengalaman bekerja agen.

education → data pendidikan terakhir agen.

Returns :

True jika data berhasil ditambahkan.

- + editAgen_Aktif(empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, empPassKey : *String*, joinDate : *Date*, position : *String*, agenStatus : *int*, phoneNo : *String*, birthPlace : *String*, birthDate : *Date*, empSex : *String*, empMail : *String*, marriageStatus : *String*, memberState : *String*, identityNo : *String*, custCareNo : *String*, bankAccount : Bank_Account_TY, workExp : Work_Exp_TY, education : Education_TY) : *Boolean*

  Melakukan pengeditan data agen yang aktif.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  empName → nama agen.

  branchNo → nomor kantor cabang tempat agen bekerja.

  address → alamat agen.

  empPassKey → password untuk login sebagai agen.

  joinDate → tanggal menjadi pegawai.

  position → posisi pegawai.

  agenStatus → status agen, aktif atau tidak aktif.

  phoneNo → nomor telepon agen.

  birthPlace → tempat lahir agen.

birthDate → tanggal lahir agen.

empSex → jenis kelamin agen.

empMail → alamat email agen.

marriageStatus → status pernikahan agen.

memberState → kewarganegaraan agen.

identityNo → nomor kartu identitas agen.

custCareNo → nomor pegawai *customer care* atasan agen.

bankAccount → data rekening bank agen.

workExp → data pengalaman bekerja agen.

education → data pendidikan terakhir agen.

Returns :

True jika data berhasil diedit.

- + editAgen_NonAktif(empNo : *String*, empName : *String*, branchNo : *String*, address : Address_TY, empPassKey : String, joinDate : *Date*, position : *String*, agenStatus : *int*, phoneNo : *String*, birthPlace : *String*, birthDate : *Date*, empSex : *String*, empMail : *String*, marriageStatus : *String*, memberState : *String*, identityNo : *String*, custCareNo : *String*, bankAccount : Bank_Account_TY, workExp : Work_Exp_TY, education : Education_TY) : *Boolean*

  Melakukan pengeditan data agen yang tidak aktif.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  empName → nama agen.

branchNo → nomor kantor cabang tempat agen bekerja.

address → alamat agen.

empPassKey → password untuk login sebagai agen.

joinDate → tanggal menjadi pegawai.

position → posisi pegawai.

agenStatus → status agen, aktif atau tidak aktif.

phoneNo → nomor telepon agen.

birthPlace → tempat lahir agen.

birthDate → tanggal lahir agen.

empSex → jenis kelamin agen.

empMail → alamat email agen.

marriageStatus → status pernikahan agen.

memberState → kewarganegaraan agen.

identityNo → nomor kartu identitas agen.

custCareNo → nomor pegawai *customer care* atasan agen.

bankAccount → data rekening bank agen.

workExp → data pengalaman bekerja agen.

education → data pendidikan terakhir agen.

Returns :

True jika data berhasil diedit.

- + deleteAgen(empNo : *String*) : *Boolean*

  Melakukan penghapusan data agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

Returns :

True jika penghapusan agen berhasil dilakukan.

- + inboxMail(empNo : *String*, mail : Mail_TY) : *Boolean*

  Melakukan penambahan data *mail* dengan status *inbox*.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Mail → data mail yang akan dimasukkan ke dalam database.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + outboxMail(empNo : *String*, *mail* : Mail_TY) : *Boolean*

  Melakukan penambahan data *mail* dengan status *outbox*.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Mail → data mail yang akan dimasukkan ke dalam database.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + addTask(empNo : *String*, task : Task_TY) : *Boolean*

  Melakukan penambahan data tugas/jadwal agen.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

Task → data tugas/jadwal agen.

Returns :

True jika penambahan data berhasil dilakukan.

- + updateTask(empNo : *String*, task : Task_TY) : *Boolean*

  Melakukan pengeditan terhadap data tugas/jadwal.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Task → data tugas/jadwal agen.

  Returns :

  True jika pengeditan data berhasil dilakukan.

- + deleteTask(empNo : *String*, task : Task_TY) : *Boolean*

  Melakukan penghapusan data tugas/jadwal.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Task → data tugas/jadwal agen.

  Returns :

  True jika penghapusan data berhasil dilakukan.

- + getInbox(empNo : *String*) : Dataset

  Mendapatkan data *mail* dengan jenis *inbox*.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Dataset yang mengandung data *mail* dengan jenis *inbox*.

- + getOutbox(empNo : *String*) : Dataset

  Mendapatkan data *mail* dengan jenis *outbox*.

  Parameters :

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  Dataset yang mengandung data *mail* dengan jenis *outbox*.

- + deleteMail_Inbox(subject : *String*, content : *String*, empNo : *String*) : *Boolean*

  Melakukan penghapusan data *mail* dengan jenis *inbox*.

  Parameters :

  Subject → subject *mail*.

  Content → isi *mail*.

  empNo → nomor pegawai yang dimiliki agen.

  Returns :

  True jika data *mail inbox* berhasil dihapus.

- + deleteMail_Outbox(subject : *String*, content : *String*, empNo : *String*) : *Boolean*

  Melakukan penghapusan data *mail* dengan jenis *outbox*.

  Parameters :

  Subject → subject *mail*.

  Content → isi *mail*.

  empNo → nomor pegawai yang dimiliki agen.

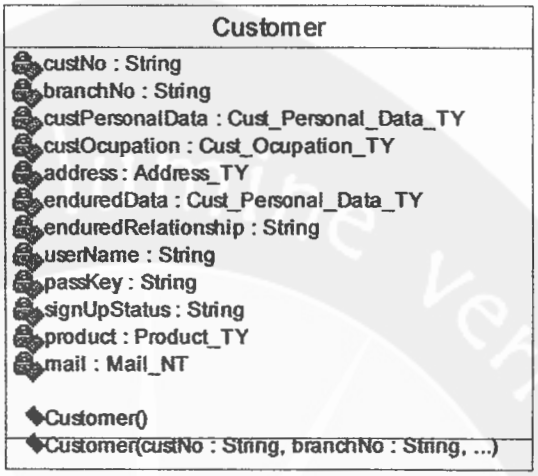
  Returns :

  True jika data *mail outbox* berhasil dihapus.

2.2.6.3. *Class CRM.Transaction.ManageAgenUI*

Lihat : *Class CRM.User.ManageAgenUI*

2.2.6.4. *Class CRM.Transaction.Customer*



Gambar 2.30 *Class CRM.Transaction.Customer*

Deskripsi

Class ini merepresentasikan *customer* dalam perangkat lunak *CRM* ini.

Atribut

- - custNo : *String*
  Merepresentasikan nomor *customer* yang dimiliki *customer*.
- - branchNo : *String*
  Merepresentasikan nomor kantor cabang tempat *customer* mendaftarkan diri.
- - custPersonalData : Cust_Personal_Data_TY
  Merepresentasikan data pribadi yang dimiliki *customer*.

- - address : Address_TY
  Merepresentasikan alamat *customer*.
- - custOcupation : Cust_Ocupation_TY
  Merepresentasikan data pekerjaan *customer*.
- - enduredData : Cust_Personal_Data_TY
  Merepresentasikan data pribadi yang dimiliki tertanggung.
- - enduredRelationship : *String*
  Merepresentasikan hubungan *customer* dengan tertanggung.
- - userName : String
  Merepresentasikan user name untuk login sebagai *customer*.
- - passKey : String
  Merepresentasikan *password* untuk login sebagai *customer*.
- - signUpStatus : *String*
  Merepresentasikan status apakah *customer* yang bersangkutan telah melakukan pendaftaran ulang atau belum.
- - product : Product_TY
  Merepresentasikan produk yang diikuti *customer*.
- - mail : Mail_NT
  Merepresentasikan daftar *mail* yang dimiliki *customer*.

Method

- + Customer()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + Customer(custPersonalData : Cust_Personal_ Data TY, custOcupation : Cust_Ocupation_TY, address : Address_TY, custPersonalDataTtg : Cust_Personal_ Data TY, custOcupationTtg : Cust_Ocupation_TY, addressTtg : Address_TY, passKey : *String*, signUpStatus : *String*, product : Product_TY, mail : Mail_NT)

  Merupakan konstruktor *class*, dengan atribut terdefinisi.

  Parameters :

  custPersonalData → data pribadi yang dimiliki *customer*.

  custOcupation → data pekerjaan yang dimiliki *customer*.

  address → alamat yang dimiliki *customer*.

  custPersonalDataTtg → data pribadi yang dimiliki oleh tertanggung.

  custOcupationTtg → data pekerjaan yang dimiliki oleh tertanggung.

  addressTtg → alamat yang dimiliki oleh tertanggung.

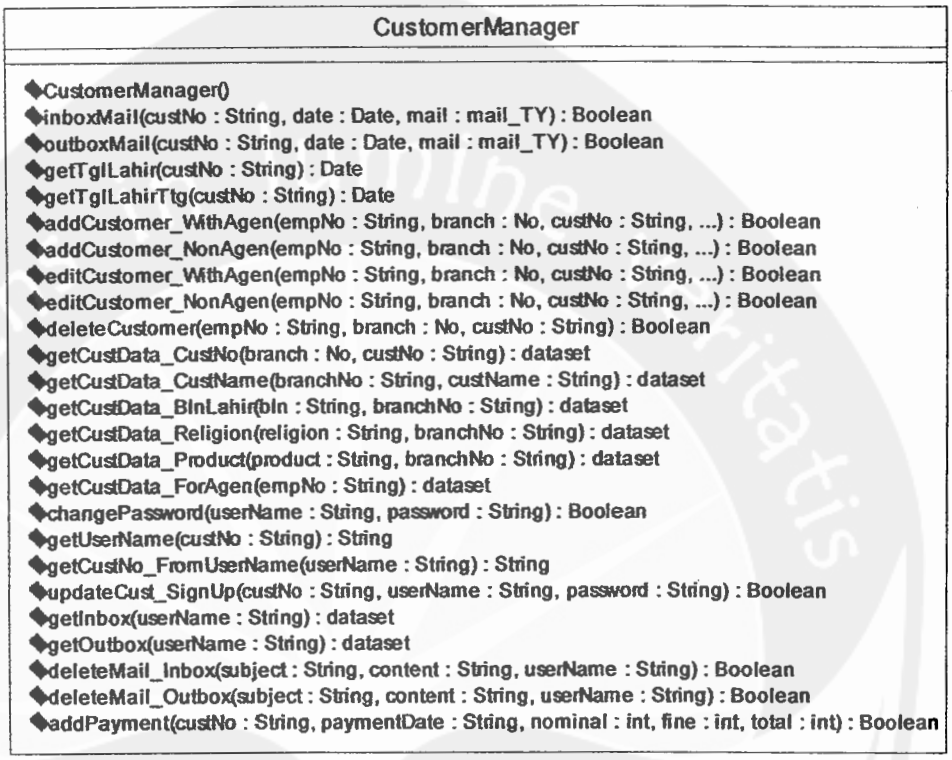
  passKey → password untuk melakukan pendaftaran pada perangkat lunak (*sing up*).

  singUpStatus → status apakah *customer* telah melakukan pendaftaran ulang atau belum.

product → data produk yang diikuti *customer*.

mail → daftar mail yang dimiliki *customer*.

2.2.6.5. *Class CRM.Transaction.CustomerManager*



Gambar 2.31 *Class CRM.Transaction.CustomerManager*

Deskripsi

Class yang berperan sebagai *control class* untuk aksi yang berhubungan dengan pengelolaan data *customer* pada *CRM*.

Atribut

-

Method

- + CustomerManager()

  Merupakan konstruksi *class*, tanpa atribut yang terdefinisi.

- + inboxMail(custNo : *String*, mail : Mail_TY) : *Boolean*

  Melakukan penambahan data *mail* dengan status *inbox*.

  Parameters :

  custNo → nomor *customer* yang dimiliki agen.

  Mail → data mail yang akan dimasukkan ke dalam database.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + outboxMail(custNo : *String*, mail : Mail_TY) : *Boolean*

  Melakukan penambahan data *mail* dengan status *outbox*.

  Parameters :

  custNo → nomor *customer* yang dimiliki agen.

  Mail → data mail yang akan dimasukkan ke dalam database.

  Returns :

  True jika penambahan data berhasil dilakukan.

- + getTglLahir(custNo : String) : Date

  Mendapatkan tanggal lahir *customer*.

  Parameters :

  custNo → nomor *customer*.

Returns :

Date (tanggal) lahir *customer*.

- + getTglLahirTtg(custNo : String) : *Date*

  Mendapatkan tanggal lahir tertanggung.

  Parameters :

  custNo → nomor tertanggung.

  Returns :

  Date (tanggal) lahir tertanggung.

- + addCustomer_WithAgen(empNo : *String*, branchNo : String, custNo : *String*, personalData : Cust_Personal_Data_TY, personalDataTtg : Cust_Personal_Data_TY, ocupation : Cust_Ocupation_TY, ocupationTtg : Cust_Ocupation_TY, address : Address_TY, adressTtg : Address_TY, relationship : String, passKey : String, product : Product_NT) : *Boolean*

  Melakukan penambahan data customer yang didapat oleh agen.

  Parameters :

  empNo → nomor pegawai (agen) yang mendapatkan *customer* tersebut.

  branchNo → nomor kantor cabang tempat *customer* mendaftarkan diri.

  custNo → nomor *customer*.

  personalData → data pribadi *customer*.

  personalDataTtg → data pribadi tertanggung.

  ocupation → data pekerjaan *customer*.

  ocupationTtg → data pekerjaan tertanggung.

address → alamat *customer*.

addressTtg → alamat tertanggung.

relationship → hubungan antara *customer* dengan tertanggung.

passKey → password yang digunakan untuk melakukan *login* sebagai *customer*.

product → data produk yang diikuti *customer*.

Returns :

True jika data *customer* berhasil ditambahkan.

- \+ addCustomer_NonAgen(empNo : *String*, branchNo : String, custNo : *String*, personalData : Cust_Personal_Data_TY, personalDataTtg : Cust_Personal_Data_TY, ocupation : Cust_Ocupation_TY, ocupationTtg : Cust_Ocupation_TY, address : Address_TY, adressTtg : Address_TY, relationship : String, passKey : String, product : Product_NT) : *Boolean*

  Melakukan penambahan data customer yang datang langsung ke kantor cabang.

  Parameters :

  empNo → nomor pegawai (*customer care*) yang mendapatkan *customer* tersebut.

  branchNo → nomor kantor cabang tempat *customer* mendaftarkan diri.

  custNo → nomor *customer*.

  personalData → data pribadi *customer*.

  personalDataTtg → data pribadi tertanggung.

  ocupation → data pekerjaan *customer*.

ocupationTtg → data pekerjaan tertanggung.

address → alamat *customer*.

addressTtg → alamat tertanggung.

relationship → hubungan antara *customer* dengan tertanggung.

passKey → password yang digunakan untuk melakukan *login* sebagai *customer*.

product → data produk yang diikuti *customer*.

Returns :

True jika data *customer* berhasil ditambahkan.

- + editCustomer_WithAgen(empNo : *String*, branchNo : String, custNo : *String*, personalData : Cust_Personal_Data_TY, personalDataTtg : Cust_Personal_Data_TY, ocupation : Cust_Ocupation_TY, ocupationTtg : Cust_Ocupation_TY, address : Address_TY, adressTtg : Address_TY, relationship : String, passKey : String, product : Product_NT) : *Boolean*

  Melakukan pengeditan data customer yang didapat oleh agen.

  Parameters :

  empNo → nomor pegawai (agen) yang mendapatkan *customer* tersebut.

  branchNo → nomor kantor cabang tempat *customer* mendaftarkan diri.

  custNo → nomor *customer*.

  personalData → data pribadi *customer*.

  personalDataTtg → data pribadi tertanggung.

ocupation → data pekerjaan *customer*.

ocupationTtg → data pekerjaan tertanggung.

address → alamat *customer*.

addressTtg → alamat tertanggung.

relationship → hubungan antara *customer* dengan tertanggung.

passKey → password yang digunakan untuk melakukan *login* sebagai *customer*.

product → data produk yang diikuti *customer*.

Returns :

True jika data *customer* berhasil diedit.

- + editCustomer_NonAgen(empNo : *String*, branchNo : String, custNo : *String*, personalData : Cust_Personal_Data_TY, personalDataTtg : Cust_Personal_Data_TY, ocupation : Cust_Ocupation_TY, ocupationTtg : Cust_Ocupation_TY, address : Address_TY, adressTtg : Address_TY, relationship : String, passKey : String, product : Product_NT) : *Boolean*

  Melakukan pengeditan data customer yang datang langsung ke kantor cabang.

  Parameters :

  empNo → nomor pegawai (agen) yang mendapatkan *customer* tersebut.

  branchNo → nomor kantor cabang tempat *customer* mendaftarkan diri.

  custNo → nomor *customer*.

  personalData → data pribadi *customer*.

personalDataTtg → data pribadi tertanggung.

ocupation → data pekerjaan *customer*.

ocupationTtg → data pekerjaan tertanggung.

address → alamat *customer*.

addressTtg → alamat tertanggung.

relationship → hubungan antara *customer* dengan tertanggung.

passKey → password yang digunakan untuk melakukan *login* sebagai *customer*.

product → data produk yang diikuti *customer*.

Returns :

True jika data *customer* berhasil diedit.

- + deleteCustomer(empNo : *String*, branchNo : String, custNo : *String*) : *Boolean*

  Melakukan penghapusan data *customer*.

  Parameters :

  empNo → nomor pegawai yang mendapatkan customer.

  branchNo → nomor kantor cabang.

  custNo → nomor *customer*.

  Returns :

  True jika data *customer* berhasil dihapus.

- + getCustData_CustNo(custNo : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data *customer* dari nomor *customer*.

  Parameters :

  custNo → nomor *customer*.

  branchNo → nomor kantor cabang.

Returns :

Dataset yang berisi data *customer.*

- + getCustData_CustName(custNo : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data customer dari nama *customer.*

  Parameters :

  custName → nama *customer.*

  branchNo → nomor kantor cabang.

  Returns :

  Dataset yang berisi data *customer.*

- + getCustData_BlnLahir(bln : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data *customer* dari bulan lahir *customer* yang bersangkutan.

  Parameters :

  bln → bulan lahir *customer.*

  branchNo → nomor kantor cabang.

  Returns :

  Dataset yang berisi data *customer.*

- + getCustData_Religion(religion : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data *customer* dari *religion customer.*

  Parameters :

  religion → agama *customer.*

  branchNo → nomor kantor cabang.

  Returns :

  Dataset yang berisi data *customer.*

- + getCustData_Product(Product : *String*, branchNo : *String*) : *Dataset*

  Mendapatkan data customer dari produk yang diikuti *customer*.

  Parameters :

  product → nomor *customer*.

  branchNo → nomor kantor cabang.

  Returns :

  Dataset yang berisi data *customer*.

- + getCustData_ForAgen(empNo : *String*) : *Dataset*

  Mendapatkan data customer dari agen yang mendapatkannya.

  Parameters :

  empNo → nomor agen yang mendapatkan *customer* yang bersangkutan.

  Returns :

  Dataset yang berisi data *customer*.

- + changePassword(userName : *String*, password : *String*) : *Boolean*

  Melakukan perubahan terhadap *password* yang *customer* miliki.

  Parameters :

  userName → *user name customer*.

  Password → *password* baru yang ingin dimasukkan.

  Returns :

  True jika data berhasil di-*update*.

- + getUserName(custNo : String) : *String*

  Mendapatkan user name yang dimiliki customer.

  Parameters :

  custNo → nomor *customer* yang ingin didapatkan *user name*-nya.

  Returns :

  String user name.

- + getCustNo_FromUserName(userName : *String*) : *String*

  Mendapatkan nomor *customer* dari *user name*.

  Parameters :

  userName → *user name* yang dimiliki *customer*.

  Returns :

  String nomor *customer*.

- + updateCust_SignUp(custNo : *String*, userName : *String*, password : *String*) : *Boolean*

  Melakukan pengeditan terhadap status *sign up*.

  Parameters :

  custNo → nomor *customer*.

  userName → *user name customer*.

  password → *password customer*.

  Returns :

  True jika pengeditan data berhasil dilakukan.

- + getInbox(userName : *String*) : Dataset

  Mendapatkan data *mail* dengan jenis *inbox*.

  Parameters :

  userName → *user name* yang dimiliki *customer*.

Returns :

Dataset yang mengandung data *mail* dengan jenis *inbox*.

- + getOutbox(userName : *String*) : Dataset

  Mendapatkan data *mail* dengan jenis *outbox*.

  Parameters :

  userName → *user name* yang dimiliki *customer*.

  Returns :

  Dataset yang mengandung data *mail* dengan jenis *outbox*.

- + deleteMail_Inbox(subject : *String*, content : *String*, userName : *String*) : *Boolean*

  Melakukan penghapusan data *mail* dengan jenis *inbox*.

  Parameters :

  Subject → subject *mail*.

  Content → isi *mail*.

  userName → *user name* yang dimiliki *customer*.

  Returns :

  True jika data *mail inbox* berhasil dihapus.

- + deleteMail_Outbox(subject : *String*, content : *String*, userName : *String*) : *Boolean*

  Melakukan penghapusan data *mail* dengan jenis *outbox*.

  Parameters :

  Subject → subject *mail*.

  Content → isi *mail*.

  userName → *user name* yang dimiliki *customer*.

Returns :

True jika data *mail outbox* berhasil dihapus.

- + addPayment(custNo : *String*, paymentDate : *String*, nominal : *int*, fine : *int*, total : *int*) : *Boolean*

  Melakukan penambahan data *payment* apabila customer melakukan pembayaran premi.

  Parameters :

  custNo → nomor *customer*.

  paymentDate → tanggal pembayaran premi.

  nominal → nominal premi yang harus dibayar.

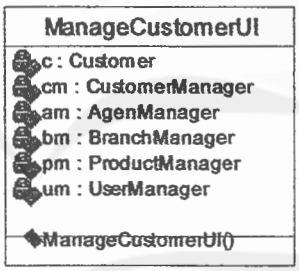
  fine → denda yang dikenakan.

  Total → total yang dibayarkan.

  Returns :

  True jika penambahan data berhasil dilakukan.

2.2.6.6. *Class CRM.Transaction.ManageCustomerUI*



Gambar 2.32 *Class CRM.Transaction.ManageCustomerUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *customer*.

Atribut

- - c : Customer

  Merepresentasikan *object control class* untuk pengelolaan *customer* saat melakukan perubahan terhadap data *customer*.

- - cm : CustomerManager

  Merepresentasikan *object control class* untuk pengelolaan *customer* saat melakukan perubahan terhadap data *customer*.

- - am : AgenManager

  Merepresentasikan *object control class* untuk pengelolaan agen saat melakukan perubahan terhadap data *customer*.

- - bm : BranchManager

  Merepresentasikan *object control class* untuk pengelolaan data kantor cabang saat melakukan perubahan terhadap data *customer*.

- - pm : ProductManager

  Merepresentasikan *object control class* untuk pengelolaan data produk saat melakukan perubahan terhadap data *customer*.

- - um : UserManager

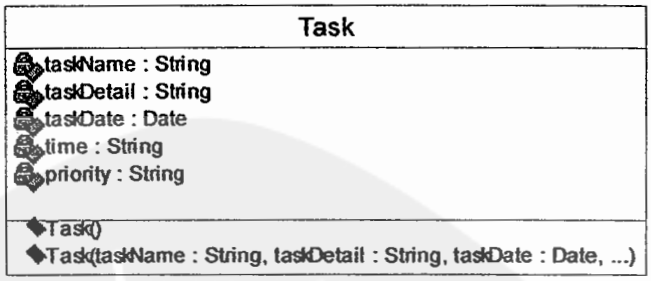
  Merepresentasikan *object control class* untuk pengelolaan *user* saat melakukan perubahan terhadap data *customer*.

Method

- + manageCustomerUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.6.7. *Class CRM.Transaction.Task*



Gambar 2.33 *Class CRM.Transaction.Task*

Deskripsi

Class ini merepresentasikan agen dalam perangkat lunak *CRM* ini.

Atribut

- - taskName : *String*

  Merepresentasikan nama tugas/jadwal yang dimiliki agen atau *customer care.*
- - taskDetail : *String*

  Merepresentasikan detail isi tugas/jadwal yang dimiliki agen atau *customer care.*
- - taskDate : *Date*

  Merepresentasikan tanggal tugas/jadwal tersebut harus dilaksanakan.
- - time : *String*

  Merepresentasikan waktu tugas/jadwal tersebut harus dilaksanakan.
- - priority : *String*

  Merepresentasikan prioritas tugas/jadwal.

Method

- + Task()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + Task (taskName : String, taskDetail : String, taskDate : Date, time : String, priority : String)

  Merupakan konstruktor *class*, dengan atribut terdefinisi.

  Parameters :

  taskName → nama tugas/jadwal.

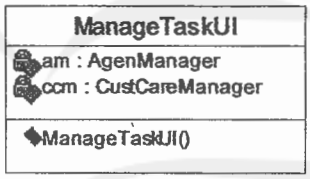
  taskDetail → detail tugas/jadwal.

  taskDate → tanggal tugas/jadwal harus dilaksanakan.

  time → waktu tugas/jadwal harus dilaksanakan.

  priority → prioritas tugas/jadwal.

2.2.6.8. *Class CRM.Transaction.ManageTaskUI*



Gambar 2.34 *Class CRM.Transaction.ManageTaskUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *task* yang dimiliki agen atau *customer care*.

Atribut

- - am : AgenManaget

  Merepresentasikan *object control class* untuk pengelolaan data agen saat melakukan perubahan terhadap data *task*.

- - ccm : CustCareManager

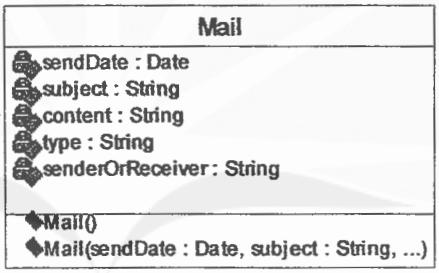
  Merepresentasikan *object control class* untuk pengelolaan data *customer care* saat melakukan perubahan terhadap data *task*.

Method

- + manageTaskUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.6.9. *Class CRM.Transaction.Mail*



Gambar 2.35 *Class CRM.Transaction.Mail*

Deskripsi

Class ini merepresentasikan agen dalam perangkat lunak *CRM* ini.

Atribut

- - sendDate : Date

  Merepresentasikan tanggal mail dikirimkan.

- - subject : *String*

  Merepresentasikan detail isi tugas/jadwal yang dimiliki agen atau *customer care*.

- - content : *Date*

  Merepresentasikan isi mail.

- - type : *String*

  Merepresentasikan tipe mail.

- - senderOrReceiver : *String*

  Merepresentasikan pengirim atau penerima mail.

Method

- + Task()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

- + Task(sendDate : Date, subject, : String, content : Strng, type : String, senderOrReceiver : String)

  Merupakan konstruktor *class*, dengan atribut terdefinisi.

  Parameters :

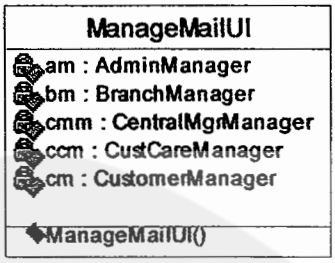
  sendDate → tanggal mail dikirim.

  subject → subyek mail.

  type → tipe email.

  senderOrReceiver → pengirim atau penerima mail.

2.2.6.10. *Class CRM.Transaction.ManageMailUI*



Gambar 2.36 *Class CRM.Transaction.ManageMailUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses pengelolaan *mail* .

Atribut

- - am : AdminManager
  Merepresentasikan *object control class* untuk pengelolaan *admin* saat melakukan perubahan terhadap data *mail*.
- - bm : BranchManager
  Merepresentasikan *object control class* untuk pengelolaan data *branch* saat melakukan perubahan terhadap data *mail*.
- - cmm : CentralMgrManager
  Merepresentasikan *object control class* untuk pengelolaan data *central manager* saat melakukan perubahan terhadap data *mail*.
- - ccm : CustCareManager
  Merepresentasikan *object control class* untuk pengelolaan data *customer care* saat melakukan perubahan terhadap data *mail*.

- - cm : CustomerManager

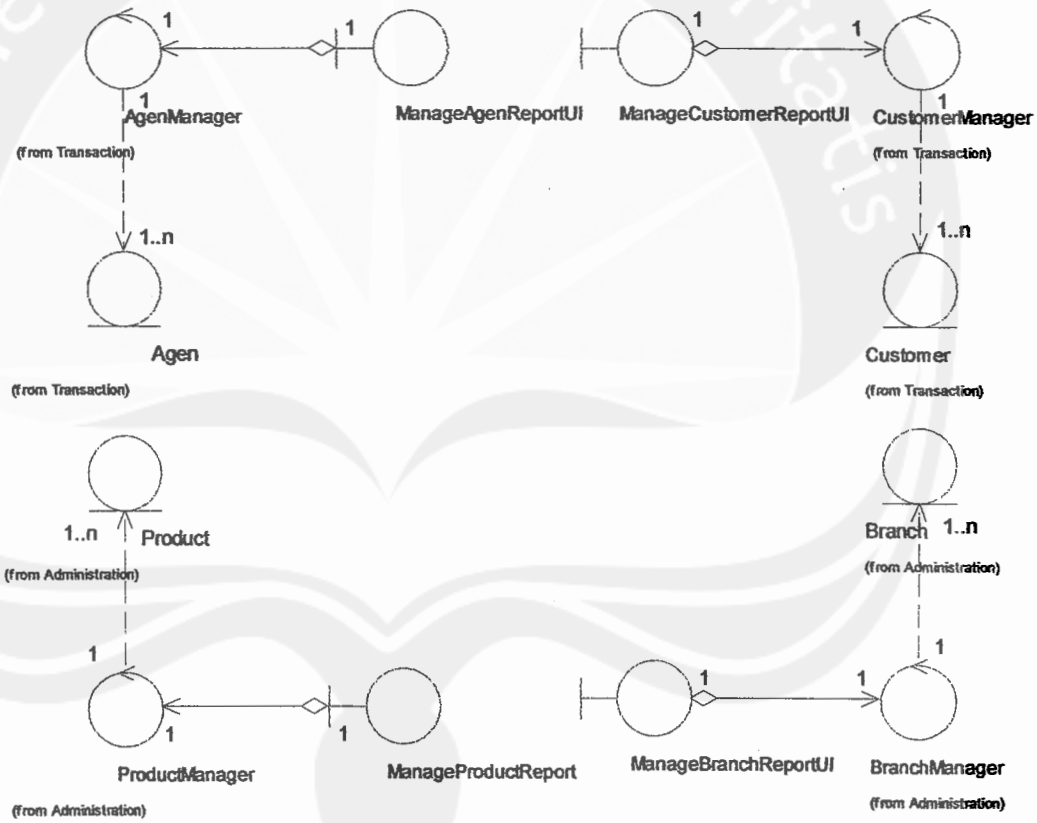
  Merepresentasikan *object control class* untuk pengelolaan data *customer* saat melakukan perubahan terhadap data *mail.*

Method

- + ManageMailUI()

  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

2.2.7. *Package Reporting*



Gambar 2.37 *Class CRM.Reporting*

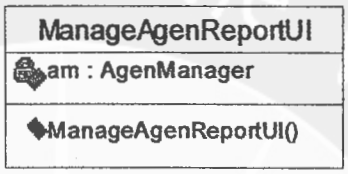
2.2.7.1. *Class CRM.Reporting.Agen*

Lihat : *Class CRM.Transaction.Agen*

2.2.7.2. *Class CRM.Reporting.AgenManager*

Lihat : *Class CRM.Transaction.AgenManager*

2.2.7.3. *Class CRM.Reporting.ManageAgenReportUI*



Gambar 2.38 *Class CRM.Reporting.ManageAgenReportUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses *reporting agen* .

Atribut

- - am : AgenManager
  Merepresentasikan *object control class* untuk pengelolaan *reporting* agen.

Method

- + ManageAgenReportUI()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

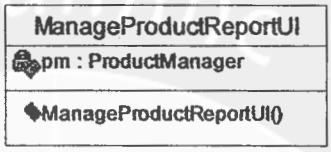
2.2.7.4. *Class CRM.Reporting.Product*

Lihat : *Class CRM.Administration.Product*

2.2.7.5. *Class CRM.Reporting.ProductManager*

Lihat : *Class CRM.Administration.ProductManager*

2.2.7.6. *Class CRM.Reporting.ManageProductReportUI*



Gambar 2.39 *Class CRM.Reporting.ManageProductReportUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses *reporting product* .

Atribut

- \- pm : ProductManager
  Merepresentasikan *object control class* untuk pengelolaan *reporting product*.

Method

- \+ ManageProductReportUI()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

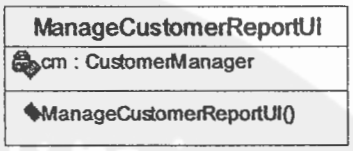
2.2.7.7. *Class CRM.Reporting.Customer*

Lihat : *Class CRM.Transaction.Customer*

2.2.7.8. *Class CRM.Reporting.CustomerManager*

Lihat : *Class CRM.Transaction.CustomerManager*

2.2.7.9. *Class CRM.Reporting.ManageCustomerReportUI*



Gambar 2.40 *Class CRM.Reporting.ManageProductReportUI*

Deskripsi

Class yang berperan sebagai *boundary class (GUI)* untuk aksi yang berhubungan dengan proses *reporting customer.*

Atribut

- - cm : CustomerManager
  Merepresentasikan *object control class* untuk pengelolaan *reporting customer.*

Method

- + ManageCustomerReportUI()
  Merupakan konstruktor *class*, tanpa atribut terdefinisi.

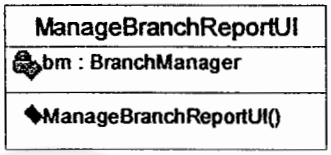
2.2.7.10. *Class CRM.Reporting.Branch*

Lihat : *Class CRM.Administration.Branch*

2.2.7.11. *Class CRM.Reporting.BranchManager*

Lihat : *Class CRM.Administration.BranchManager*

2.2.7.12. *Class CRM.Reporting.ManageBranchReportUI*



Gambar 2.41 *Class CRM.Reporting.ManageProductReportUI*

Deskripsi

Class yang berperan sebagai *boundary class* (*GUI*) untuk aksi yang berhubungan dengan proses *reporting branch* (kantor cabang).

Atribut

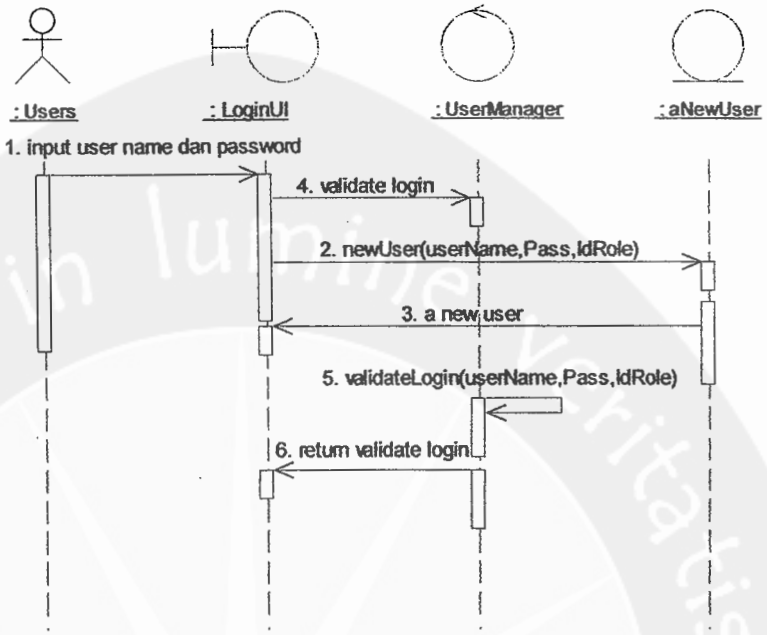
- \- bm : BranchManager
  Merepresentasikan *object control class* untuk pengelolaan *reporting branch* (kantor cabang).

Method

- \+ ManageBranchReportUI()
  Merupakan konstruktor class, tanpa atribut terdefinisi.

2.3. Realisasi *Use Case*

2.3.1. *Use Case : Login*



Gambar 2.42 Realisasi *Use Case : Login*

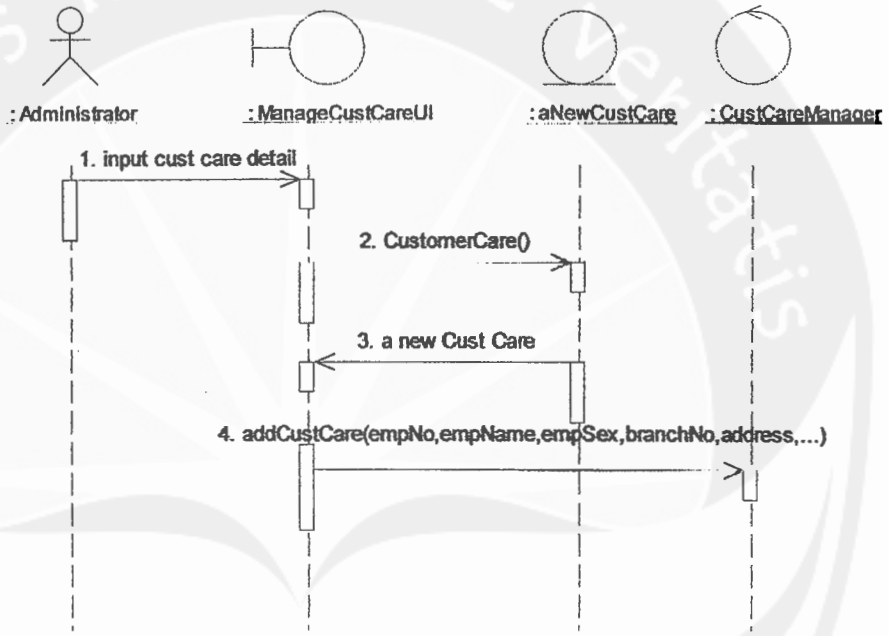
Flow of events :

1. *User* menampilkan antarmuka untuk *login* yaitu *boundary class* LoginUI, lalu memasukkan *user name* dan *password* untuk melakukan *login*.
2. LoginUI akan melakukan *create instance user*.
3. *Instance user* yang baru akan dikembalikan ke *control class* UserManager.
4. LoginUI akan memanggil fungsi untuk melakukan valiadasi *login*, yang dilakukan di dalam *control class* UserManager.

5. *Control class* UserManager akan melakukan validasi *login* (melakukan pengecekan *user name*, *password*, dan hak akses yang dimiliki).
6. *Control class* UserManager mengembalikan hasil validasi ke *boundary class* LoginUI.

2.3.2. *Use Case Manage Customer Care*

2.3.2.1. *Use Case : Add Customer Care*



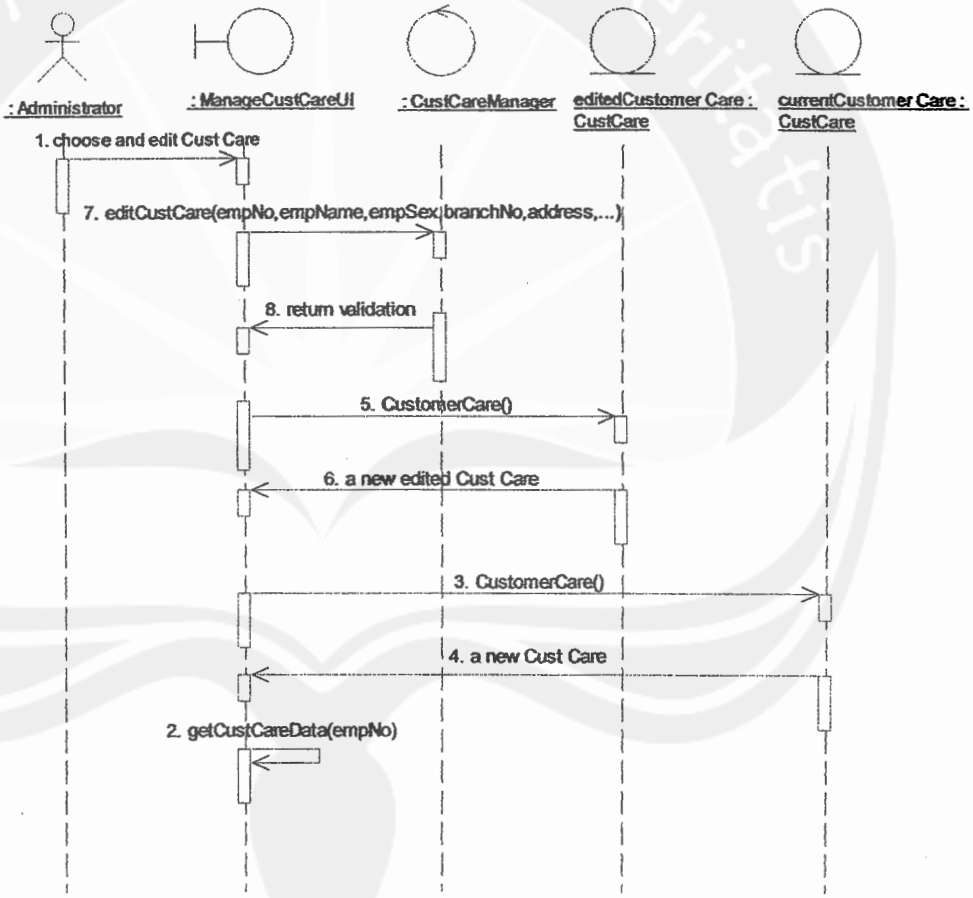
Gambar 2.43 Realisasi *Use Case : Add Customer Care*

Flow of events :

1. *User* (administrator) menampilkan antarmuka untuk melakukan penambahan *customer care* yaitu *boundary class* ManageCustCareUI, lalu memasukkan data detail *customer care* yang baru.

2. ManageCustCareUI akan melakukan *create instance customer care*.
3. *Instance customer care* yang baru akan dikembalikan ke *boundary class* ManageCustCareUI.
4. Apabila detail *customer care* baru yang dimasukkan *user valid*, maka data *customer care* baru tersebut akan ditambahkan ke dalam *database* oleh *control class* CustCareManager.

2.3.2.2. *Use Case : Edit Customer Care*

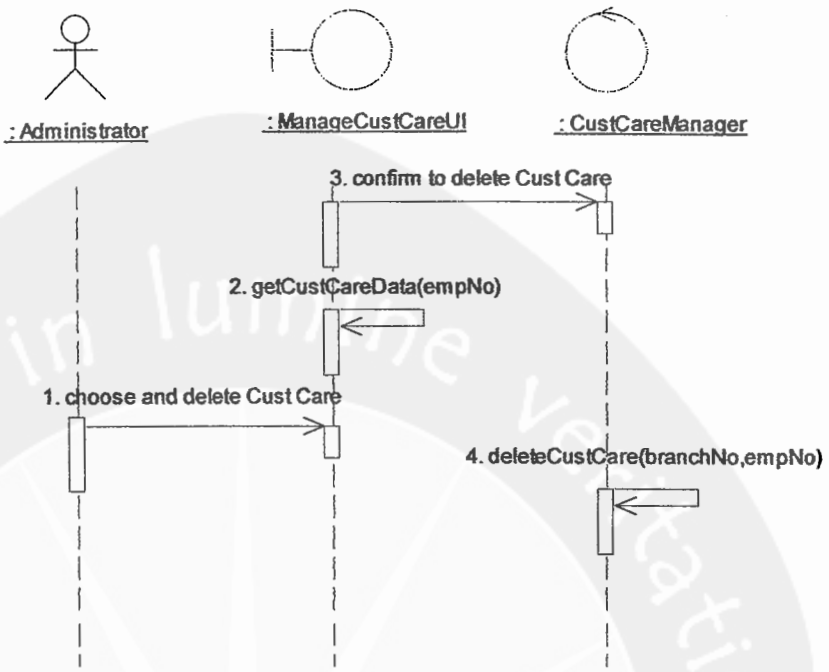


Gambar 2.44 Realisasi *Use Case : Edit Customer Care*

Flow of events :

1. *User* memilih dan melakukan pengeditan terhadap data *customer care.*
2. *Boundary class* ManagerCustCareUI menampilkan data *customer care* yang dipilih *user* dan hendak diedit.
3. ManageCustCareUI akan melakukan *create instance customer care.*
4. *Instance customer care* yang baru akan dikembalikan ke *boundary class* ManageCustCareUI.
5. ManageCustCareUI akan melakukan *create instance edited customer care* untuk data *customer care* yang telah diedit.
6. *Instance edited customer care* yang baru akan dikembalikan ke *boundary class* ManageCustCareUI.
7. Perubahan terhadap data *customer care* akan di-*update* ke dalam *database* oleh *control class* CustCareManager.
8. Apabila *update* ke dalam *database* berhasil dilakukan, *validation* bernilai *true* akan dikembalikan.

2.3.2.3. *Use Case : Delete Customer Care*

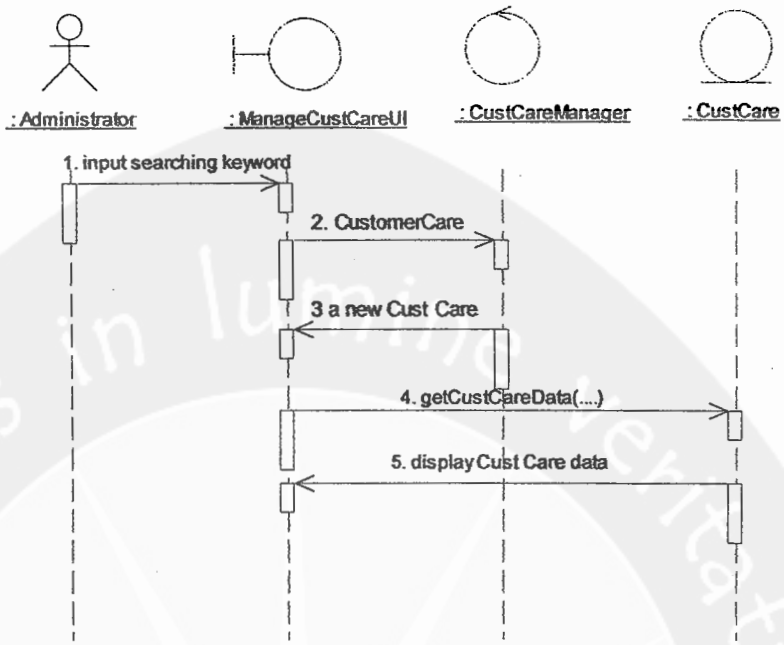


Gambar 2.45 Realisasi *Use Case : Delete Customer Care*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *customer care*.
2. *Boundary Class* ManageCustCareUI menampilkan data *customer care* yang ingin dihapus.
3. *Boundary Class* ManageCustCareUI meminta konfirmasi penghapusan dan akan dikirim ke *control class* CustCareManager.
4. *Control class* CustCareManager akan melakukan penghapusan dari database.

2.3.2.4. *Use Case : Search Customer Care*



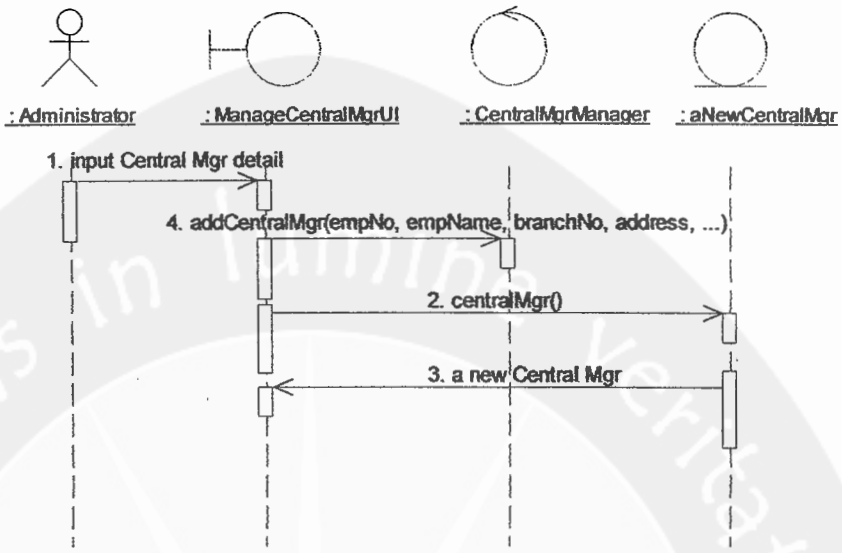
Gambar 2.46 Realisasi *Use Case : Search Customer Care*

Flow of events :

1. *User* memasukkan *keyword* pencarian *customer care.*
2. *Boundary class* ManageCustCareUI melakukan *create instance customer care.*
3. *Instance customer care* yang baru akan dikembalikan ke *boundary class* ManageCustCareUI.
4. *control class* CustCare akan melakukan validasi *keyword* dan mendapatkan data *customer care.*
5. Data *customer care* ditampilkan pada *boundary class* ManageCustCareUI.

2.3.3. *Use Case Manage Central Manager*

2.3.3.1. *Use Case : Add Central Manager*

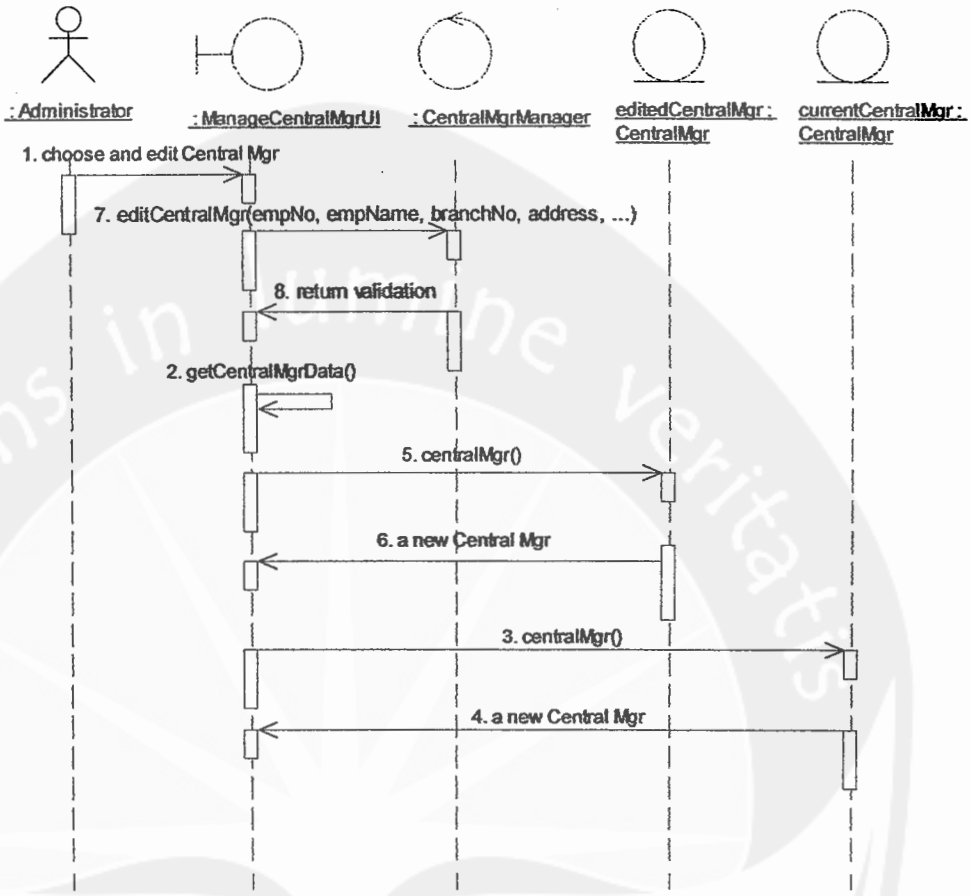


Gambar 2.46 Realisasi *Use Case : Add Central Manager*

Flow of events :

1. *User* memasukkan detail *central manager* yang ingin ditambahkan.
2. *Boundary class* ManageCentralMgrUI melakukan *create instance central manager.*
3. *Instance central manager* yang baru akan dikembalikan ke *boundary class* ManageCentralMgrUI.
4. *User* mengkonfirmasi penambahan *central manager* dan *control class* CentralMgrManager akan menyimpannya ke dalam *database*.

2.3.3.2. *Use Case : Edit Central Manager*



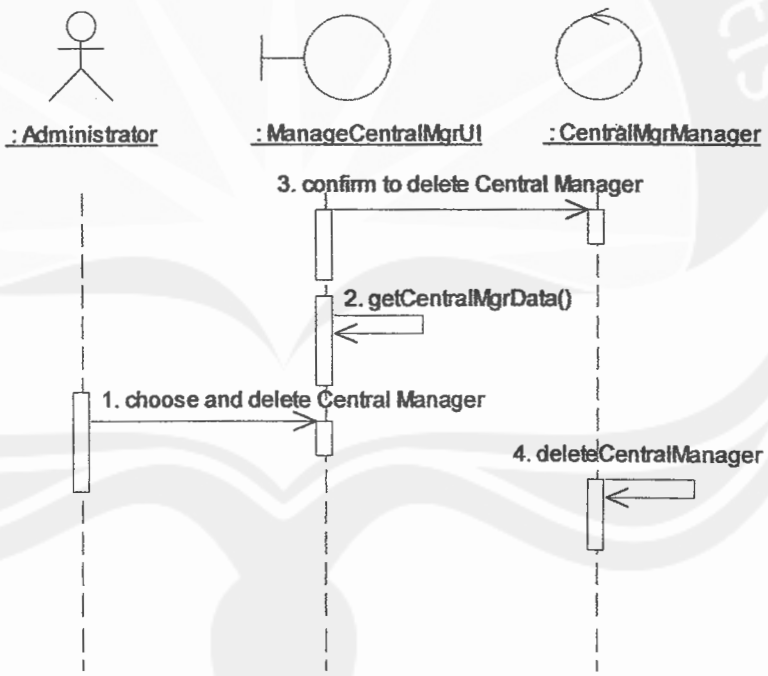
Gambar 2.47 Realisasi *Use Case : Edit Central Manager*

Flow of events :

1. *User* memilih *central manager* yang ingin dedit.
2. *Boundary class* ManageCentralMgrUI menampilkan data *central manager*.
3. *Boundary class* ManageCentralMgrUI melakukan *create instance central manager*.

4. *Instance central manager* yang baru akan dikembalikan ke *boundary class* ManageCentralMgrUI.
5. *Boundary class* ManageCentralMgrUI melakukan *create instance edited central manager.*
5. *Instance edited central manager* yang baru akan dikembalikan ke *boundary class* ManageCentralMgrUI.
6. *Control class* CentralMgrManager melakukan penyimpanan ke dalam database.
7. Apabila data baru berhasil disimpan, *control class* CentralMgrManager akan mengembailkan nilai yang *valid.*

2.3.3.3. *Use Case : Delete Central Manager*

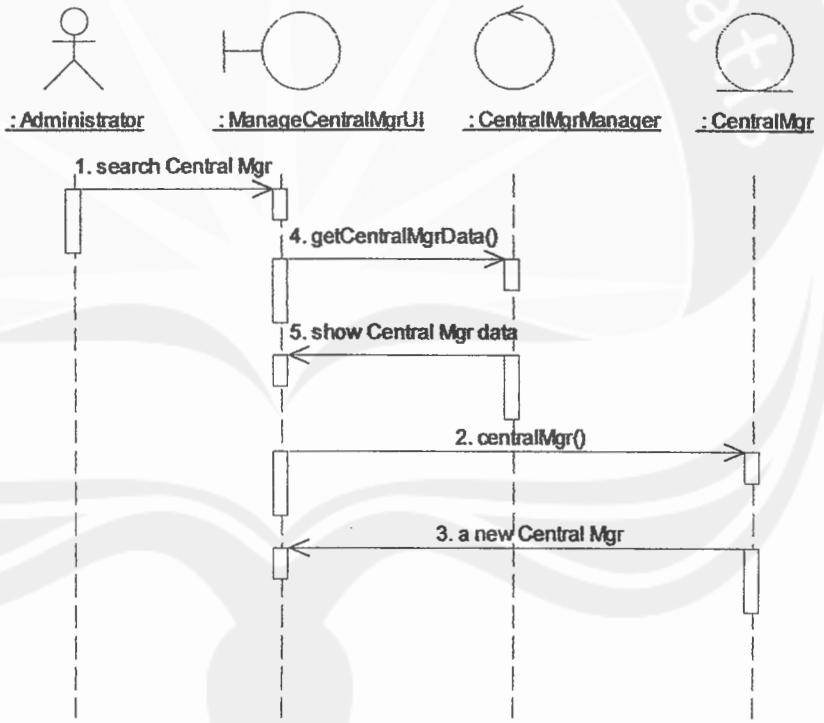


Gambar 2.48 Realisasi *Use Case : Delete Central Manager*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *central manager.*
2. *Boundary Class* ManageCentralMgrUI menampilkan data *central manager* yang ingin dihapus.
3. *Boundary Class* ManageCentralMgrUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* CentralMgrManager.
4. *Control class* CentralMgrManager akan melakukan penghapusan dari *database*.

2.3.3.4. *Use Case : Display Central Manager*



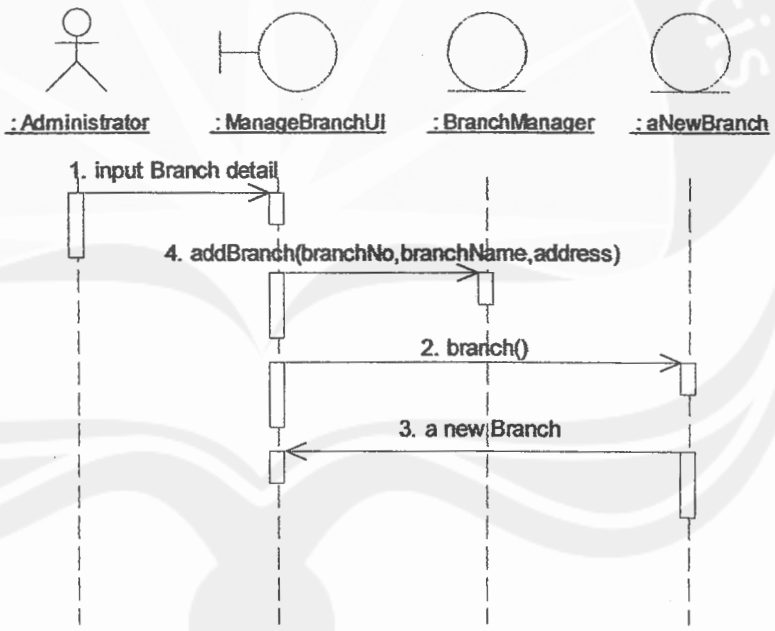
Gambar 2.49 Realisasi *Use Case : Display Central Manager*

Flow of events :

1. *User* melakukan pencarin *central manager.*
2. *Boundary class* ManageCentralMgrUI melakukan *create instance central manager.*
3. *Instance central manager* yang baru akan dikembalikan ke *boundary class* ManageCentralMgrUI.
4. *Boundary class* ManageCentralMgrUI akan memanggil *control class* CentralMgrManager untuk menampilkan data *central manager.*
5. Data *central manager* akan tertampilkan.

2.3.4. *Use Case Manage Branch*

2.3.4.1. *Use Case : Add Branch*

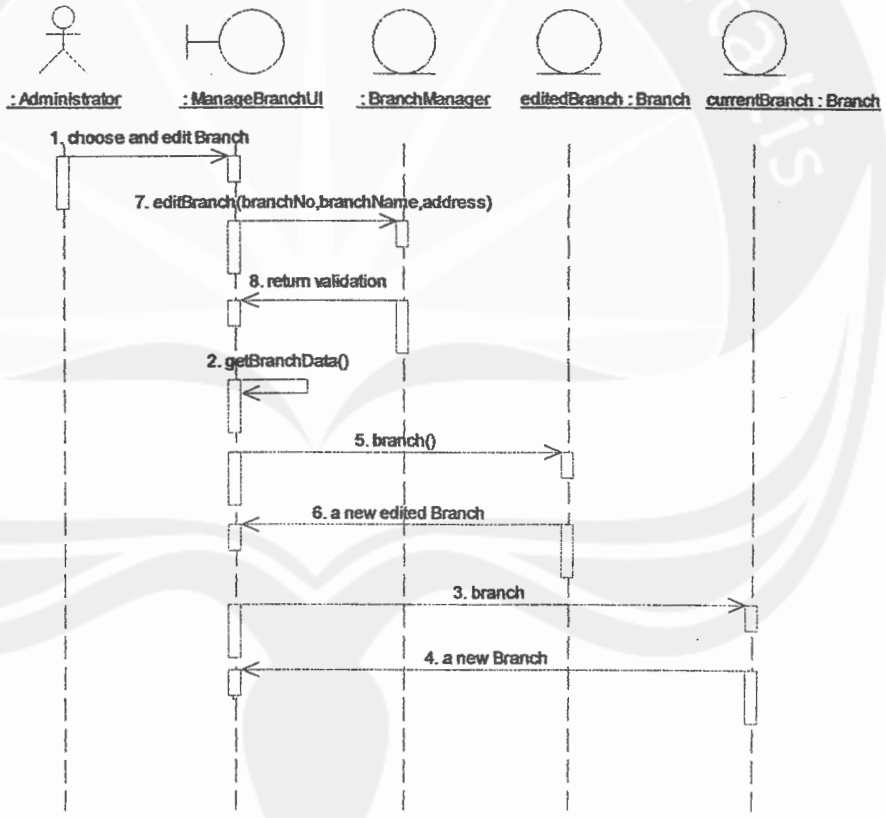


Gambar 2.50 Realisasi Use Case : Display Central Manager

Flow of events :

1. *User* memasukkan detail *branch* (kantor cabang) yang ingin ditambahkan.
2. *Boundary class* ManageBranchUI melakukan *create instance branch.*
3. *Instance branch* yang baru akan dikembalikan ke boundary *class* ManageBranchUI.
4. *User* mengkonfirmasi penambahan *branch* dan *control class* BranchManager akan menyimpannya ke dalam *database.*

2.3.4.2. *Use Case : Edit Branch*

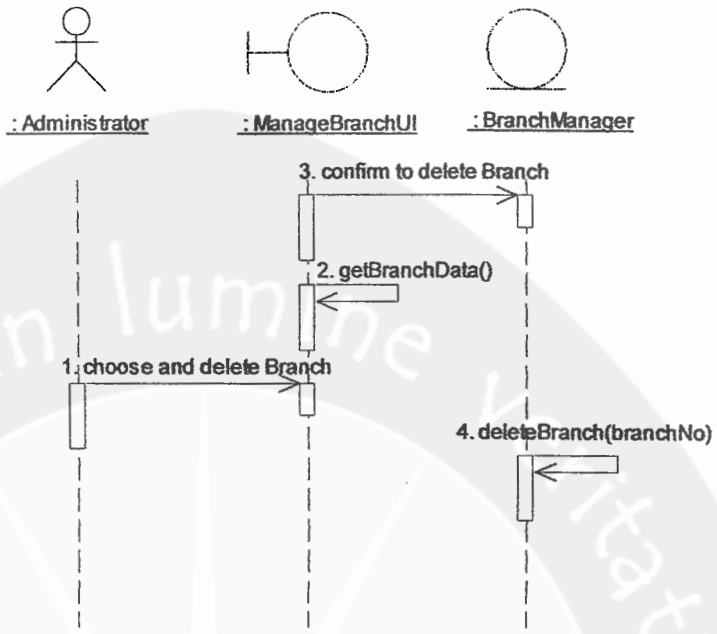


Gambar 2.51 Realisasi *Use Case : Edit Branch*

Flow of events :

1. *User* memilih *branch* yang ingin dedit.
2. *Boundary class* ManageBranchUI menampilkan data *branch*.
3. *Boundary class* ManageBranchUI melakukan *create instance branch*.
4. *Instance branch* yang baru akan dikembalikan ke *boundary class* ManageBranchUI.
5. *Boundary class* ManageBranchUI melakukan *create instance edited branch*.
6. *Instance edited branch* yang baru akan dikembalikan ke *boundary class* ManageBranchUI.
7. *Control class* BranchManager melakukan penyimpanan ke dalam *database*.
8. Apabila *data* baru berhasil disimpan, *control class* BranchManager akan mengembailkan nilai yang *valid*.

2.3.4.3. *Use Case : Delete Branch*



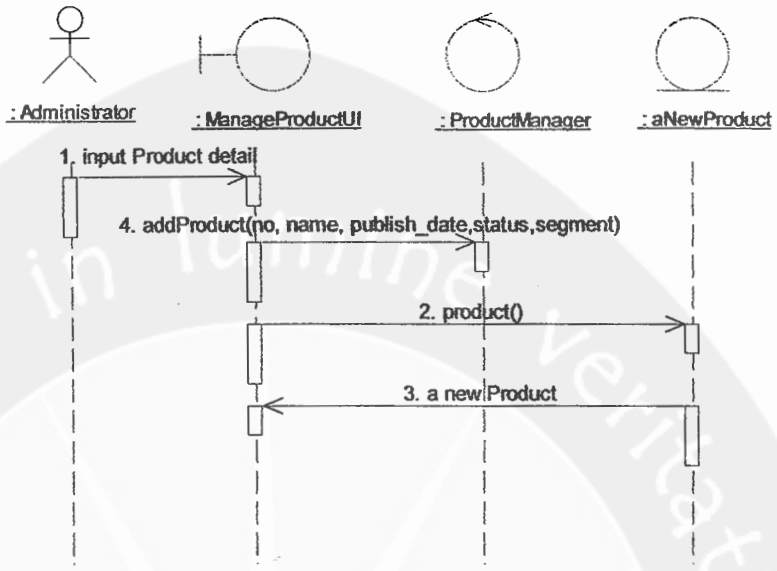
Gambar 2.52 Realisasi *Use Case : Delete Branch*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *branch*.
2. *Boundary Class* ManageBranchUI menampilkan data *branch* yang ingin dihapus.
3. *Boundary Class* ManageBranchUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* BranchManager.
4. *Control class* BranchManager akan melakukan penghapusan dari *database*.

2.3.5. *Use Case Manage Product*

2.3.5.1. *Use Case : Add Product*

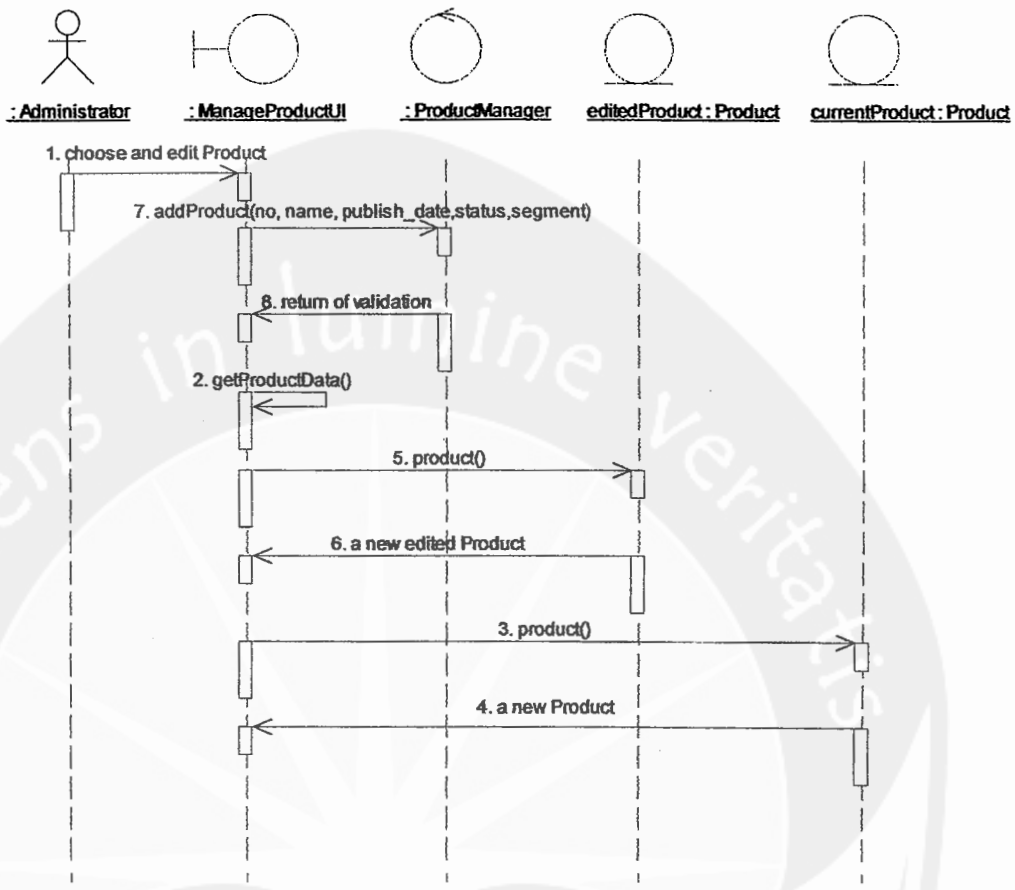


Gambar 2.53 Realisasi *Use Case : Add Product*

Flow of events :

1. *User* memasukkan detail *product* yang ingin ditambahkan.
2. *Boundary class* ManageProductUI melakukan *create* instance *product*.
3. *Instance branch* yang baru akan dikembalikan ke *boundary class* ManageProductUI.
4. *User* mengkonfirmasi penambahan *product* dan *control class ProductManager* akan menyimpannya ke dalam *database*.

2.3.5.2. *Use Case : Edit Product*



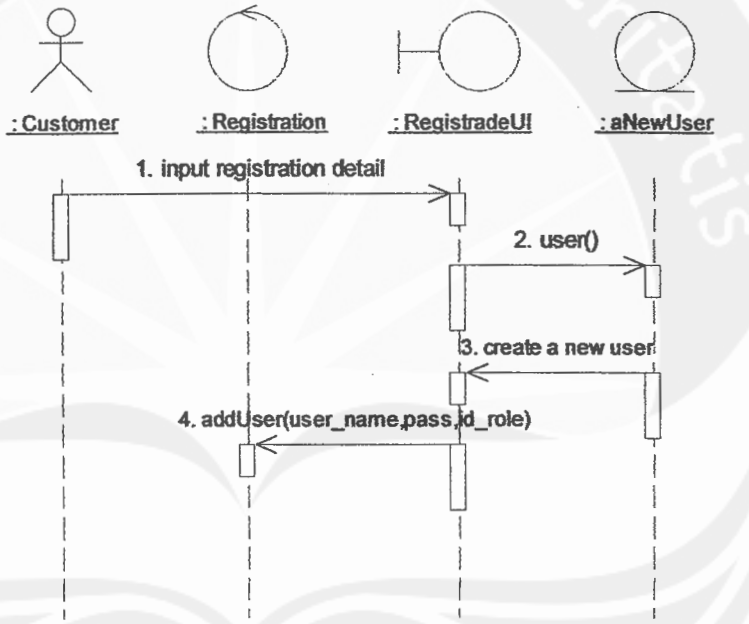
Gambar 2.54 Realisasi *Use Case : Edit Product*

Flow of events :

1. *User* memilih *product* yang ingin diedit.
2. *Boundary class* ManageProductUI menampilkan data *product*.
3. *Boundary class* ManageProductUI melakukan *create instance product*.
4. *Instance branch* yang baru akan dikembalikan ke *boundary class* ManageProductUI.

5. *Boundary class* ManageProductUI melakukan *create instance edited product.*
6. *Instance edited branch* yang baru akan dikembalikan ke *boundary class* ManageProductUI.
7. *Control class* ProductManager melakukan penyimpanan ke dalam *database.*
8. Apabila *data* baru berhasil disimpan, *control class* ProductManager akan mengembailkan nilai yang *valid.*

2.3.6. *Use Case Registration*



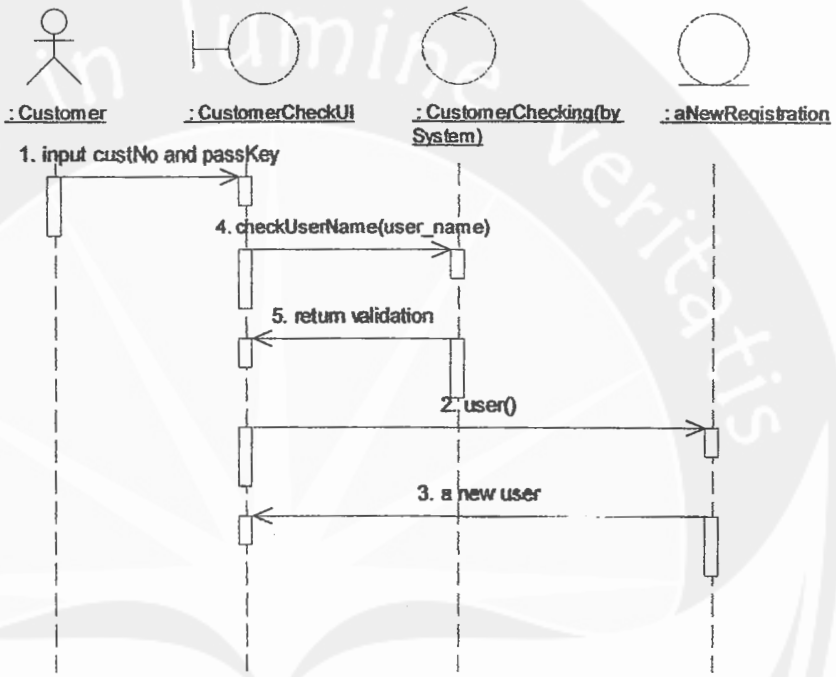
Gambar 2.55 Realisasi *Use Case : Registration*

Flow of events :

1. *User* memasukkan data *detail registration.*
2. *Boundary class* RegistrationUI melakukan *create instance user.*

3. *Instance user* yang baru akan dikembalikan ke *boundary class* RegistrationUI.
4. *Control class* Registration akan melakukan penyimpanan registrasi ke dalam *database*.

2.3.7. *Use Case Customer Check*



Gambar 2.56 Realisasi *Use Case : Customer Check*

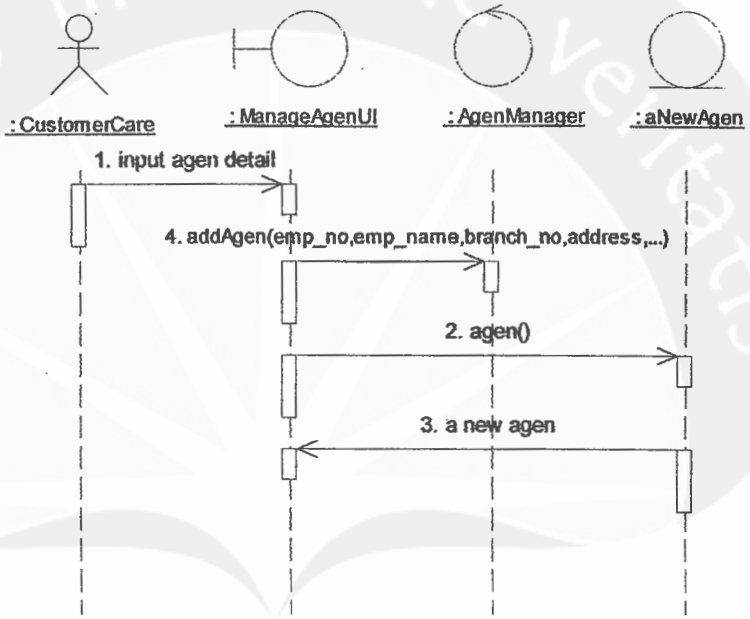
Flow of events :

1. *User* memasukkan customer number dan pass key untuk melakukan *customer check*.
2. *Boundary class* CustomerCheckUI melakukan *create instance registration*.
3. *Instance registration* yang baru akan dikembalikan ke *boundary class* CustomerCheckUI.

4. *Control class* CustomerChecking akan melakukan pengecekan *customer number* dan *pass key* ke dalam *database*.
5. *Control class* CustomerChecking akan mengembalikan nilai validasi customer.

2.3.8. *Use Case Manage Agen*

2.3.8.1. *Use Case : Add Agen*



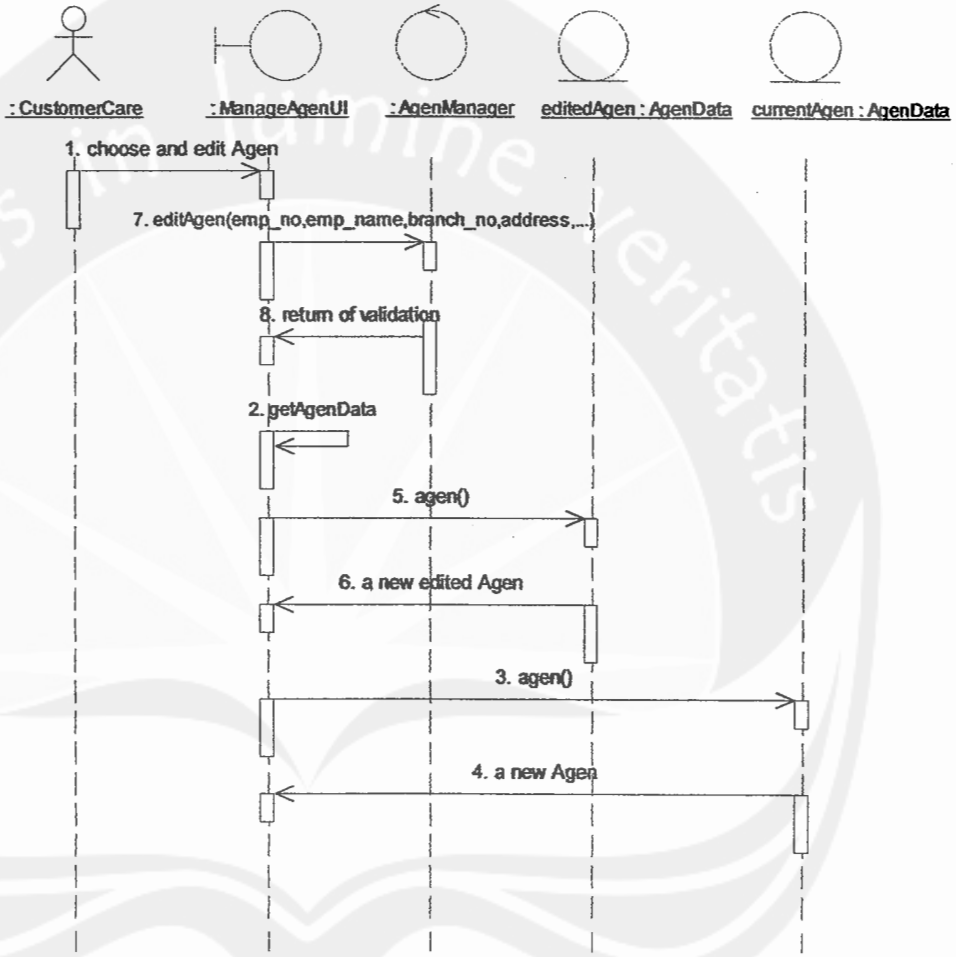
Gambar 2.57 Realisasi *Use Case : Add Agen*

Flow of events :

1. *User* memasukkan detail agen yang ingin ditambahkan.
2. *Boundary class* ManageAgenUI melakukan *create instance* agen.
3. *Instance* agen yang baru akan dikembalikan ke *boundary class* ManageAgenUI.

4. *User* mengkonfirmasi penambahan agen dan *control class* AgenManager akan menyimpannya ke dalam *database*.

2.3.8.2. *Use Case : Edit Agen*



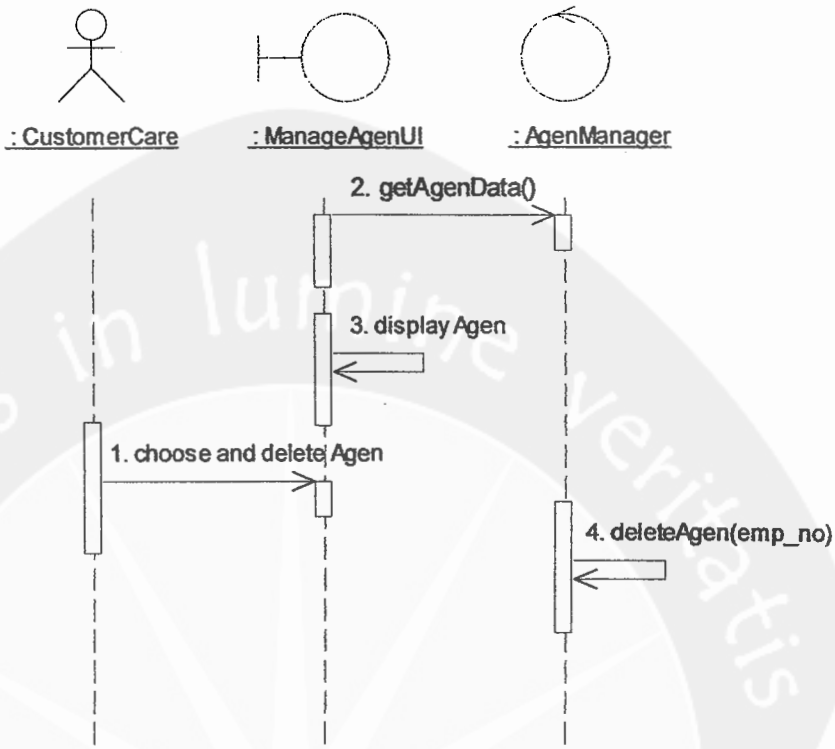
Gambar 2.58 Realisasi Use Case : Edit Agen

Flow of events :

1. *User* memilih agen yang ingin diedit.
2. *Boundary class* ManageAgenUI menampilkan data agen.

3. *Boundary class* ManageAgenUI melakukan *create instance* agen.
4. *Instance* agen yang baru akan dikembalikan ke *boundary class* ManageAgenUI.
5. *Boundary class* ManageAgenUI melakukan *create instance edited* agen.
6. *Instance edited* agen yang baru akan dikembalikan ke *boundary class* ManageAgenUI.
7. *Control class* AgenManager melakukan penyimpanan ke dalam *database*.
8. Apabila *data* baru berhasil disimpan, *control class* AgenManager akan mengembailkan nilai yang *valid*.

2.3.8.3. *Use Case : Delete Agen*

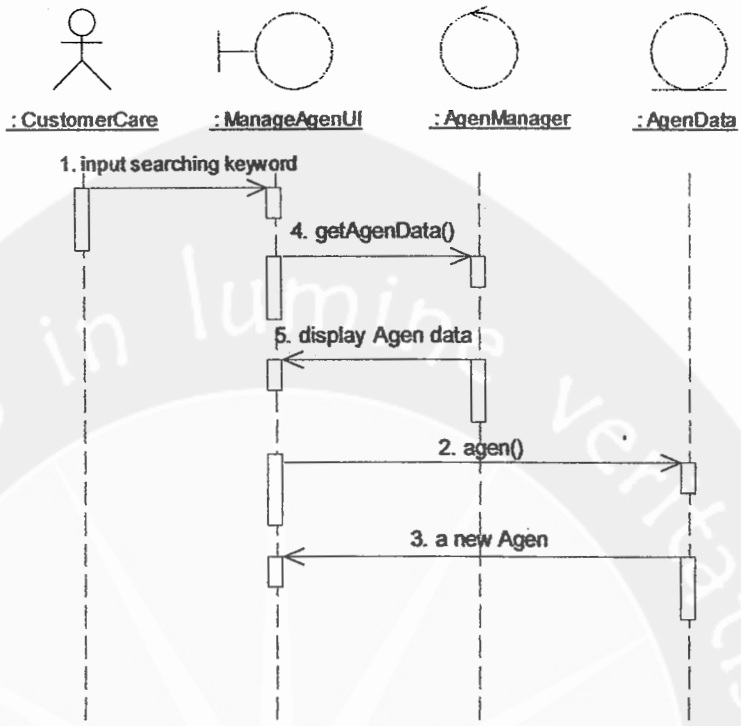


Gambar 2.59 Realisasi *Use Case : Delete Agen*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data agen.
2. *Boundary Class* ManageAgenUI menampilkan data agen yang ingin dihapus.
3. *Boundary Class* ManageAgenUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* AgenManager.
4. *Control class* AgenManager akan melakukan penghapusan dari *database.*

2.3.8.4. *Use Case : Search Agen*



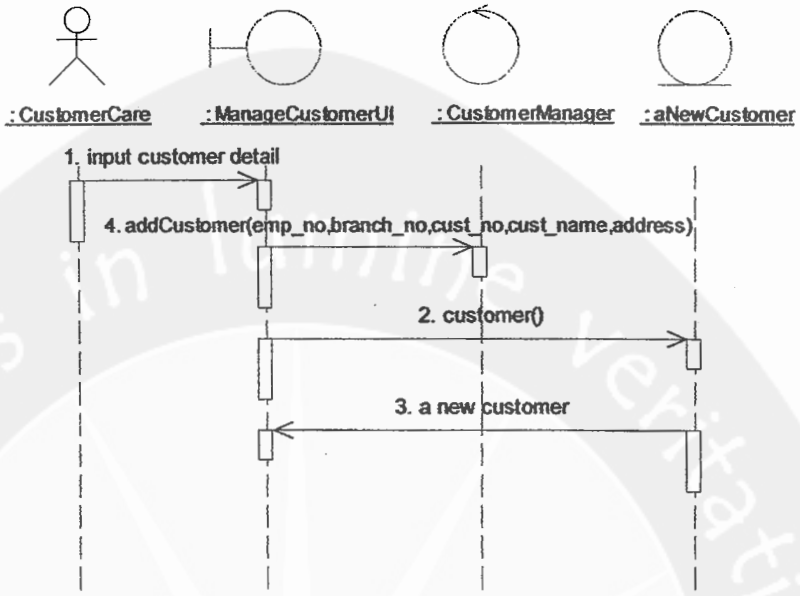
Gambar 2.60 Realisasi *Use Case : Search Agen*

Flow of events :

1. *User* memasukkan *keyword* pencarian agen.
2. *Boundary class* ManageAgenUI melakukan *create instance* agen.
3. *Instance* agen yang baru akan dikembalikan ke *boundary class* ManageAgenUI.
4. *control class* AgenManager akan melakukan validasi *keyword* dan mendapatkan data agen.
5. Data agen ditampilkan pada *boundary class* ManageAgenUI.

2.3.9. *Use Case Manage Customer*

2.3.9.1. *Use Case : Add Customer*

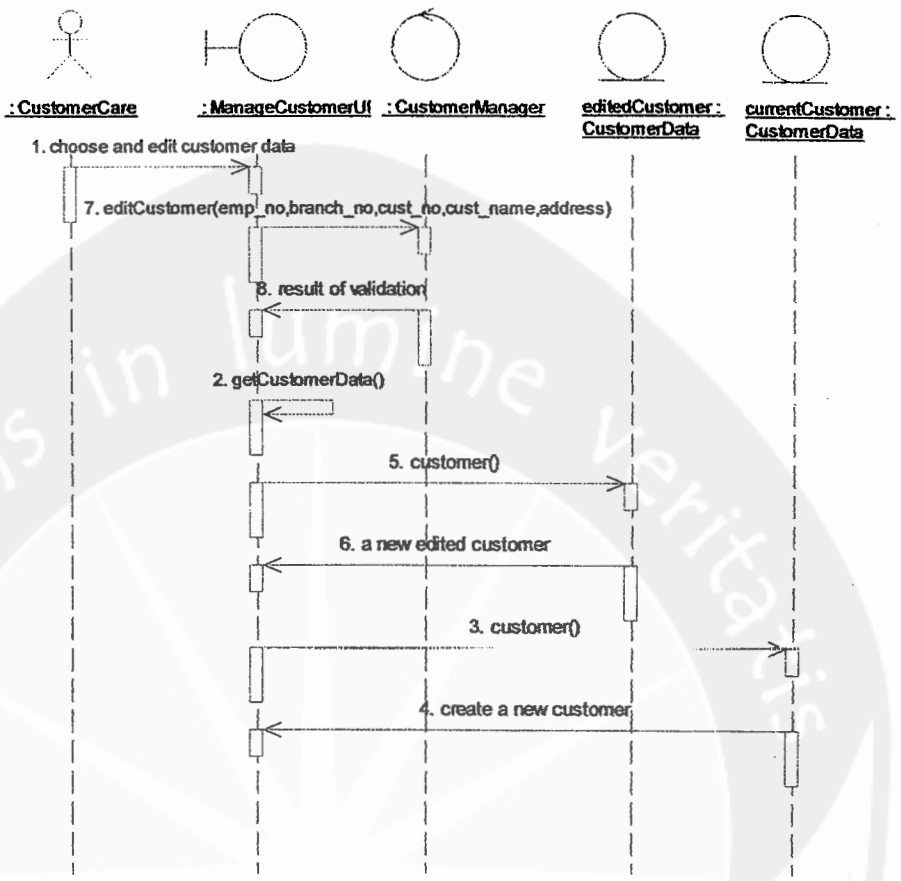


Gambar 2.61 Realisasi Use Case : Add Customer

Flow of events :

1. *User* memasukkan detail *customer* yang ingin ditambahkan.
2. *Boundary class* ManageCustomerUI melakukan *create* instance *customer*.
3. *Instance customer* yang baru akan dikembalikan ke *boundary class* ManageCustomerUI.
4. *User* mengkonfirmasi penambahan *customer* dan *control class* CustomerManager akan menyimpannya ke dalam *database*.

2.3.9.2. *Use Case : Edit Customer*



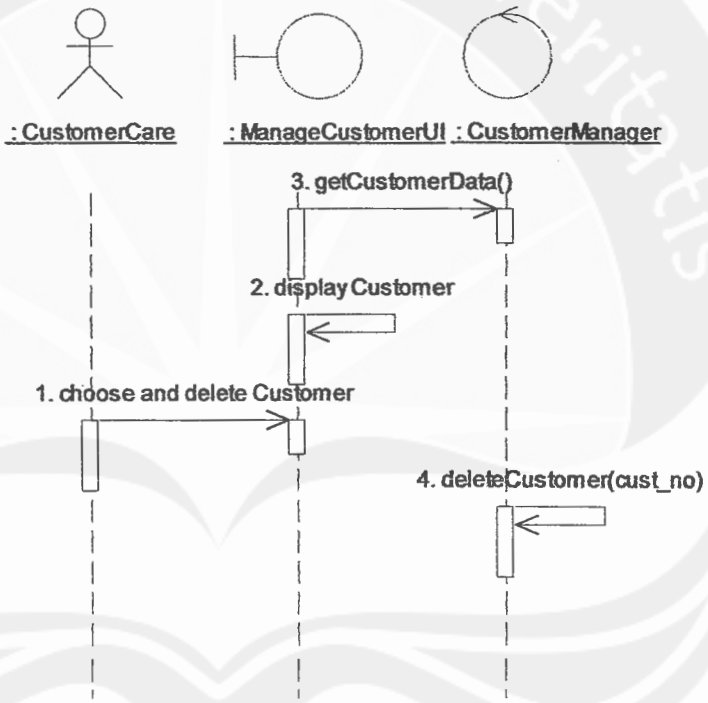
Gambar 2.62 Realisasi *Use Case : Edit Customer*

Flow of events :

1. *User* memilih *customer* yang ingin diedit.
2. *Boundary class* ManageCustomerUI menampilkan data *customer.*
3. *Boundary class* ManageCustomerUI melakukan *create instance customer.*
4. *Instance customer* yang baru akan dikembalikan ke *boundary class* ManageCustomerUI.

5. *Boundary class* ManageCustomerUI melakukan *create instance edited customer.*
6. *Instance edited customer* yang baru akan dikembalikan ke *boundary class* ManageCustomerUI.
7. *Control class* CustomerManager melakukan penyimpanan ke dalam *database.*
8. Apabila *data* baru berhasil disimpan, *control class* CustomerManager akan mengembailkan nilai yang *valid.*

2.3.9.3. *Use Case : Delete Customer*



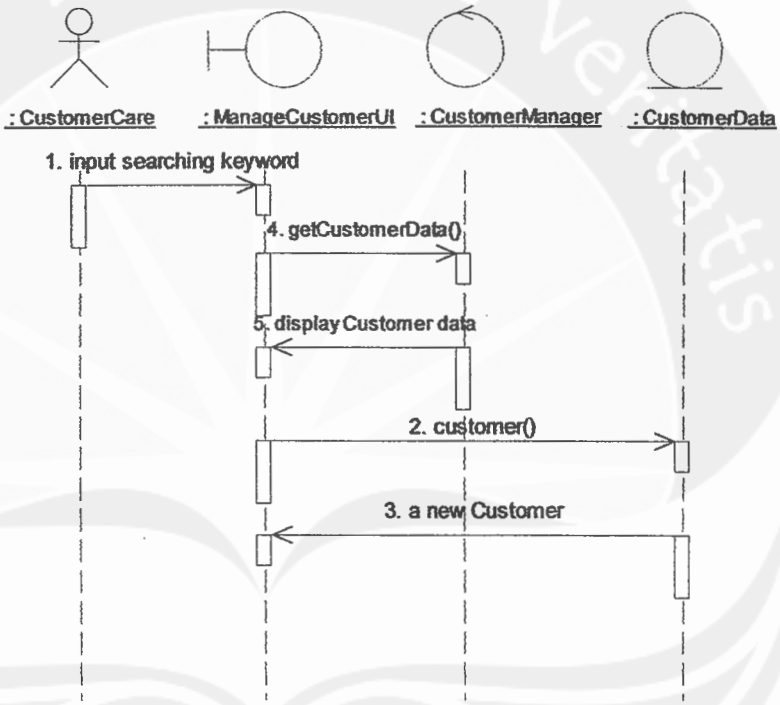
Gambar 2.63 Realisasi *Use Case : Delete Agen*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *customer.*

2. *Boundary Class* ManageCustomerUI menampilkan data *customer* yang ingin dihapus.
3. *Boundary Class* ManageCustomerUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* CustomerManager.
4. *Control class* CustomerManager akan melakukan penghapusan dari *database*.

2.3.9.4. *Use Case : Search Customer*



Gambar 2.64 Realisasi *Use Case : Search Customer*

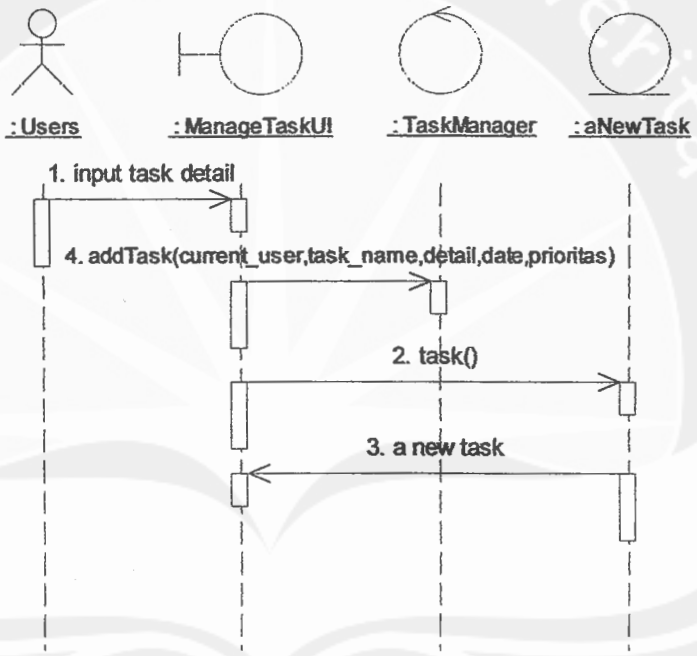
Flow of events :

1. *User* memasukkan *keyword* pencarian *customer*.
2. *Boundary class* ManageCustomerUI melakukan *create instance customer*.

3. *Instance customer* yang baru akan dikembalikan ke *boundary class* ManageCustomerUI.
4. *control class* CustomerManager akan melakukan validasi *keyword* dan mendapatkan data *customer*.
5. Data *customer* ditampilkan pada *boundary class* ManageCustomerUI

2.3.10. *Use Case Manage Task*

2.3.10.1. *Use Case : Add Task*



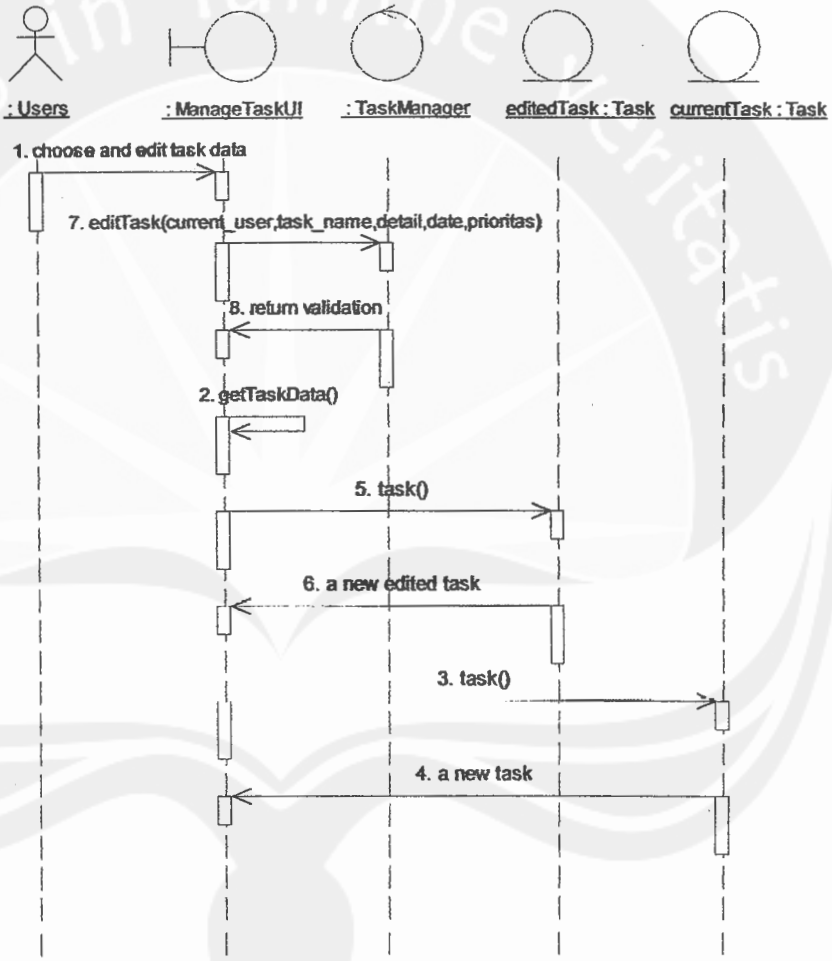
Gambar 2.65 Realisasi *Use Case : Add Task*

Flow of events :

1. *User* memasukkan detail *task* yang ingin ditambahkan.
2. *Boundary class* ManageTaskUI melakukan *create* instance *task*.

3. *Instance task* yang baru akan dikembalikan ke *boundary class* ManageTaskUI.
4. *User* mengkonfirmasi penambahan *task* dan *control class* TaskManager akan menyimpannya ke dalam *database*.

2.3.10.2. *Use Case : Edit Task*

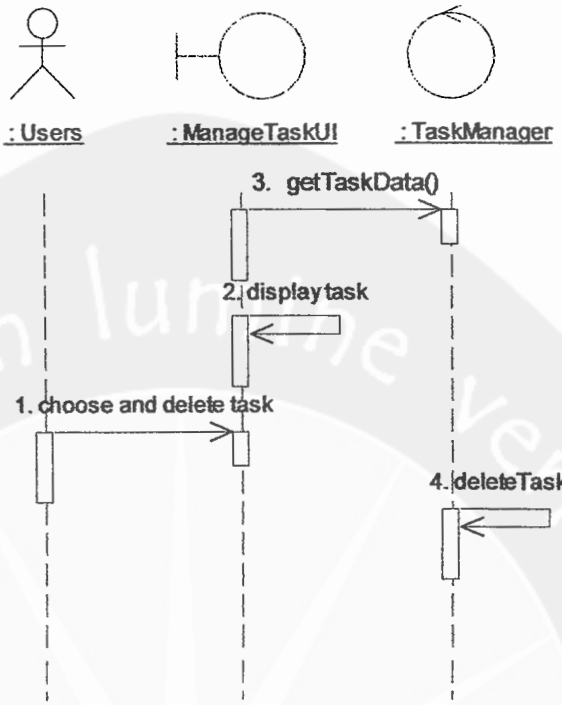


Gambar 2.66 Realisasi *Use Case : Edit Task*

Flow of events :

1. *User* memilih *task* yang ingin diedit.
2. *Boundary class* ManageTaskUI menampilkan data *task*.
3. *Boundary class* ManageTaskUI melakukan *create instance task*.
4. *Instance customer* yang baru akan dikembalikan ke *boundary class* ManageTaskUI.
5. *Boundary class* ManageTaskUI melakukan *create instance edited task*.
6. *Instance edited task* yang baru akan dikembalikan ke *boundary class* ManageTaskUI.
7. *Control class* TaskManager melakukan penyimpanan ke dalam *database*.
8. Apabila *data* baru berhasil disimpan, *control class* TaskManager akan mengembailkan nilai yang *valid*.

2.3.10.3. *Use Case : Delete Task*

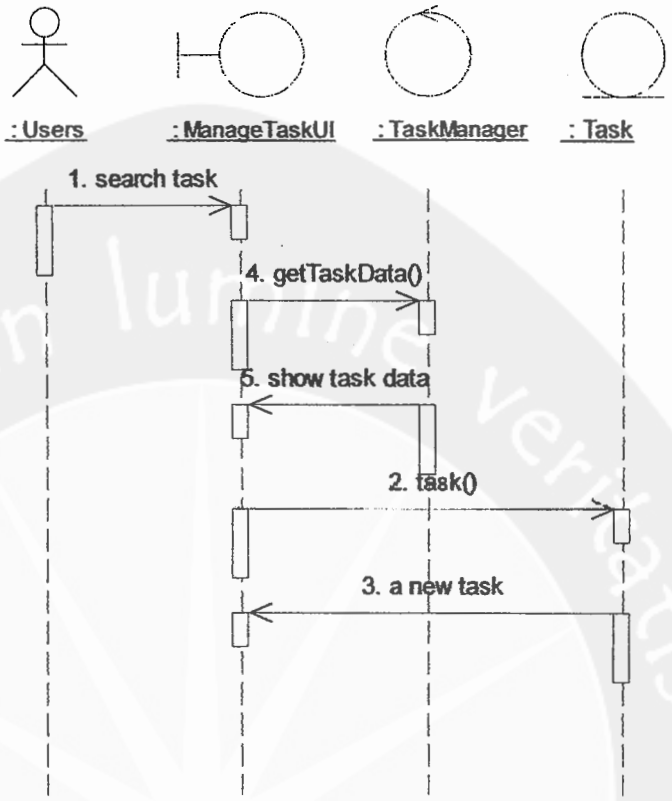


Gambar 2.67 Realisasi *Use Case : Delete Task*

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *customer.*
2. *Boundary Class* ManageCustomerUI menampilkan data *customer* yang ingin dihapus.
3. *Boundary Class* ManageCustomerUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* CustomerManager.
4. *Control class* CustomerManager akan melakukan penghapusan dari *database*.

2.3.10.4. *Use Case : Display Task*



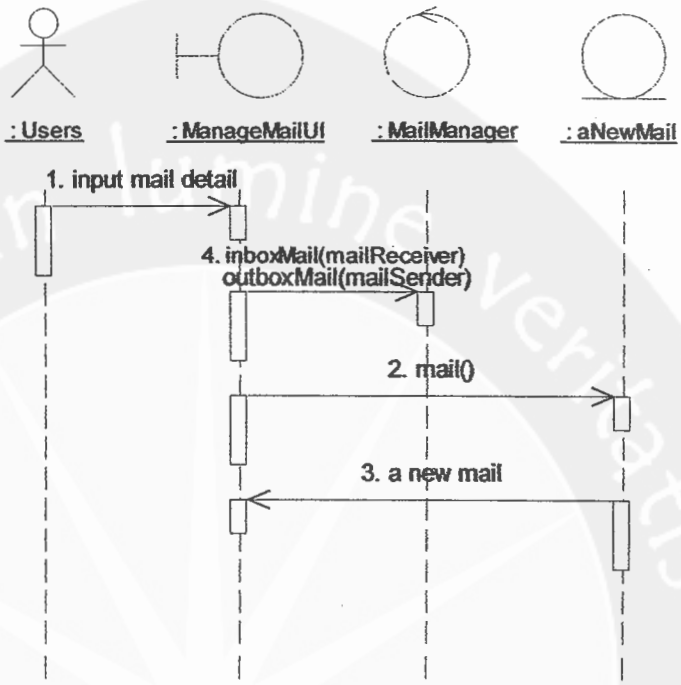
Gambar 2.68 Realisasi *Use Case : Display Task*

Flow of events :

1. *User* melakukan pencarin *task.*
2. *Boundary class* ManageTaskUI melakukan *create instance task.*
3. *Instance task* yang baru akan dikembalikan ke *boundary class* ManageTaskUI.
4. *Boundary class* ManageTaskUI akan memanggil *control class* TaskManager untuk menampilkan data *task.*
5. Data *task* akan tertampilkan.

2.3.11. *Use Case Manage Mail*

2.3.11.1. *Use Case : Create Mail*

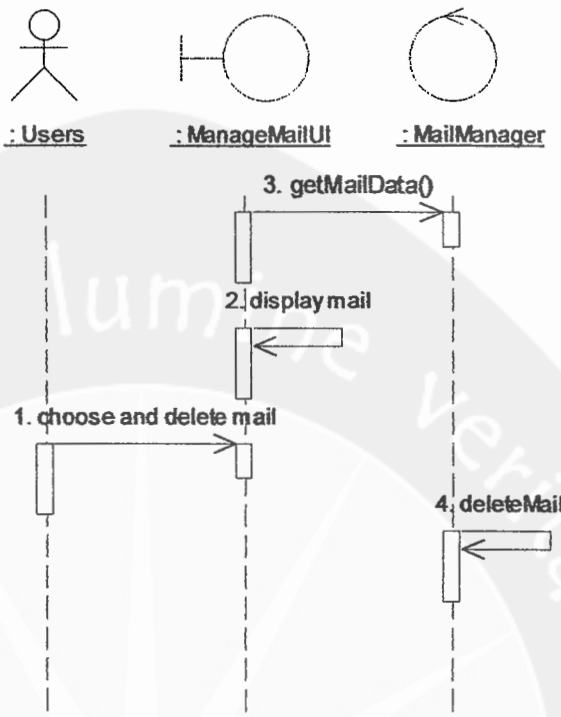


Gambar 2.69 Realisasi *Use Case : Create Mail*

Flow of events :

1. *User* memasukkan detail *mail* yang ingin ditambahkan.
2. *Boundary class* ManageMailUI melakukan *create* instance *mail*.
3. *Instance mail* yang baru akan dikembalikan ke *boundary class* ManageMailUI.
4. *User* mengkonfirmasi penambahan *mail* dan *control class* MailManager akan menyimpannya ke dalam *database*.

2.3.11.2. *Use Case : Delete Mail*

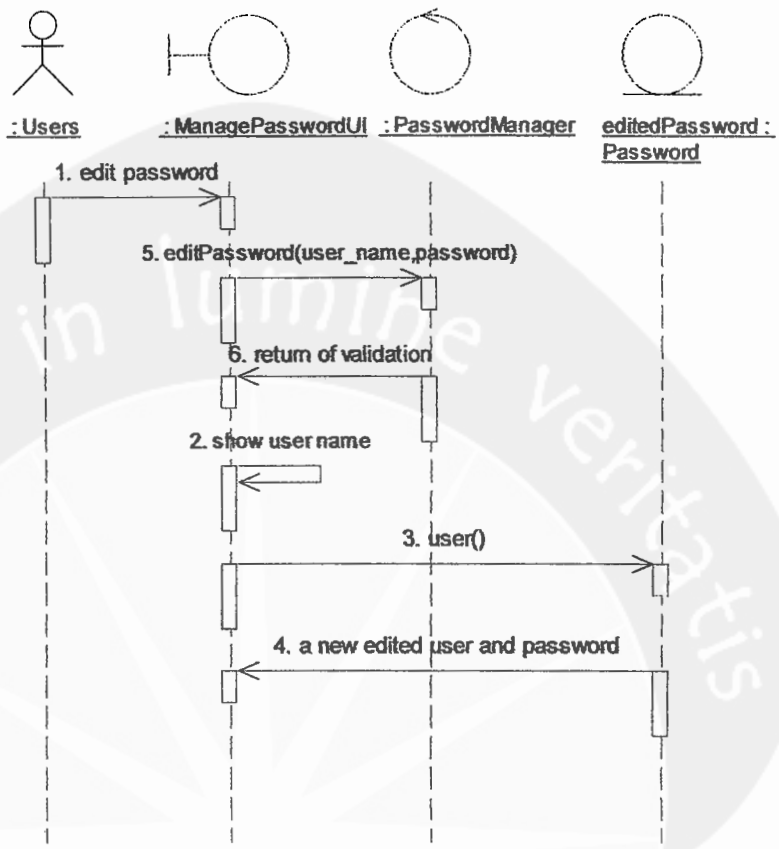


Gambar 2.70 Realisasi Use Case : Delete Mail

Flow of events :

1. *User* memilih dan melakukan penghapusan terhadap data *mail*.
2. *Boundary Class* ManageMailUI menampilkan data *mail* yang ingin dihapus.
3. *Boundary Class* ManageMailUI meminta konfirmasi *penghapusan* dan akan dikirim ke *control class* MailManager.
4. *Control class* MailManager akan melakukan penghapusan dari *database*.

2.3.12. *Use Case Manage Password*



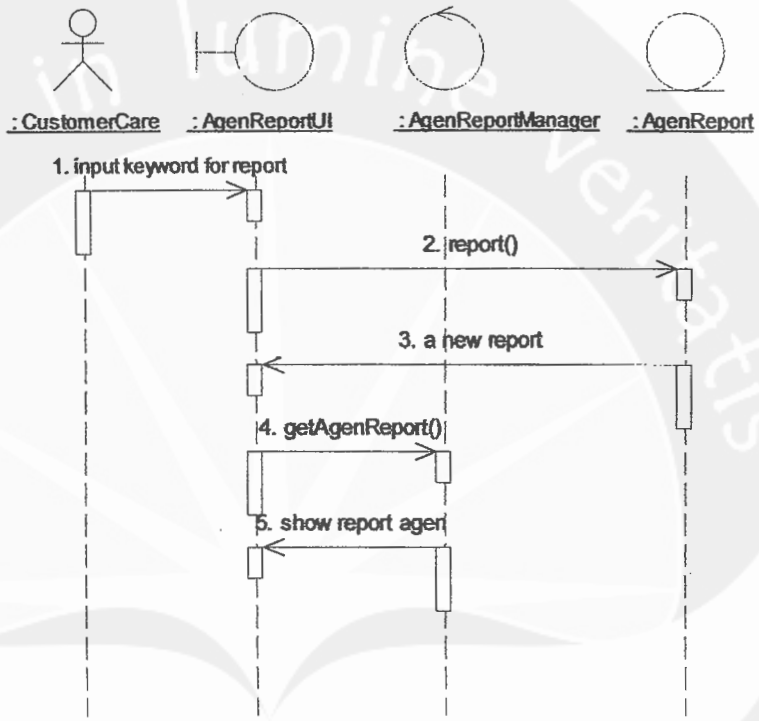
Gambar 2.71 Realisasi *Use Case : Manage Password*

Flow of events :

1. *User* masuk ke dalam *account* yang ingin diedit dan melakukan edit *password*.
2. *Boundary class* ManagePasswordUI menampilkan data *account*.
3. *Boundary class* ManagePasswordUI melakukan *create instance password*.
4. *Instance edited password* yang baru akan dikembalikan ke *boundary class* ManagePasswordUI.

5. *Control class* PasswordManager melakukan penyimpanan ke dalam *database*.
6. Apabila perubahan berhasil disimpan, *control class* PasswordManager akan mengembailkan nilai yang *valid*.

2.3.13. *Use Case Agen Report*



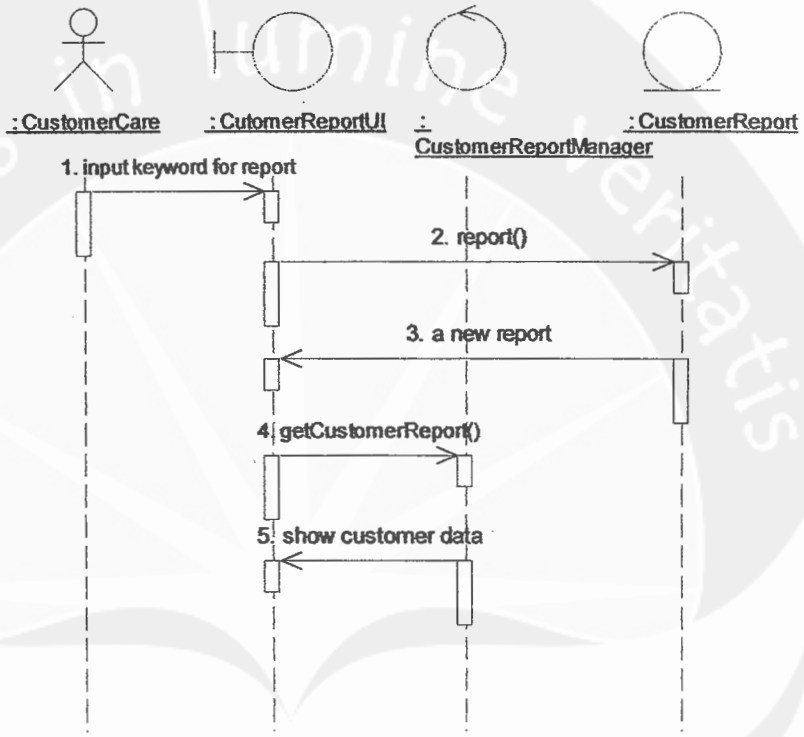
Gambar 2.72 Realisasi *Use Case : Agen Report*

Flow of events :

1. *User* memasukkan *keyword* untuk menentukan *report* yang ditampilkan.
2. *Control class* AgenReportManager melakukan *create* instance *report*.

3. *Instance report* yang baru akan dikembalikan ke *boundary class* AgenReportUI.
4. *Boundary class* akan mendapatkan data agen.
5. *Control class* akan menampilkan *report* agen.

2.3.14. *Use Case Customer Report*



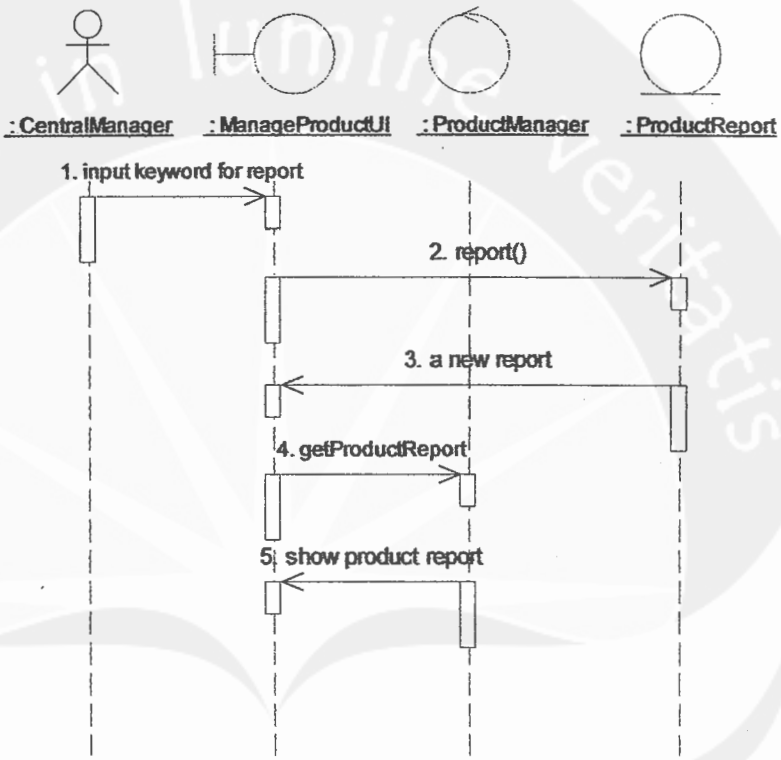
Gambar 2.73 Realisasi *Use Case : Customer Report*

Flow of events :

1. *User* memasukkan *keyword* untuk menentukan *report* yang ditampilkan.
2. *Control class* CustomerReportManager melakukan *create* instance *report*.

3. *Instance report* yang baru akan dikembalikan ke *boundary class* CustomerReportUI.
4. *Boundary class* akan mendapatkan data *customer*.
5. *Control class* akan menampilkan *report customer*.

2.3.15. *Use Case Product Report*



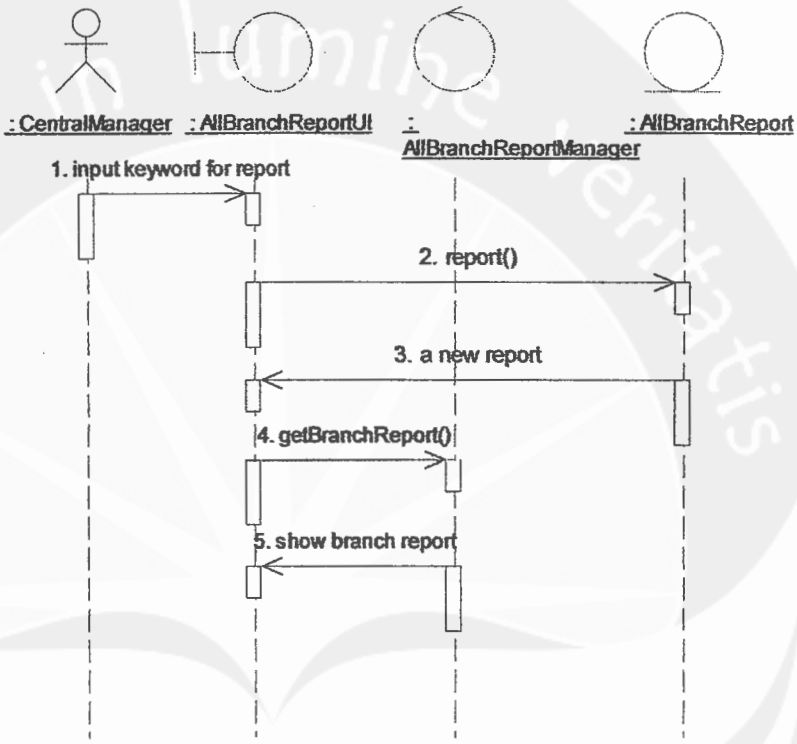
Gambar 2.74 Realisasi *Use Case : Product Report*

Flow of events :

1. *User* memasukkan *keyword* untuk menentukan *report* yang ditampilkan.
2. *Control class* ProductReportManager melakukan *create instance report*.

3. *Instance report* yang baru akan dikembalikan ke *boundary class* ProductReportUI.
4. *Boundary class* akan mendapatkan data *product.*
5. *Control class* akan menampilkan *report product.*

2.3.16. *Use Case All Branch Report*



Gambar 2.75 Realisasi *Use Case : All Branch Report*

Flow of events :

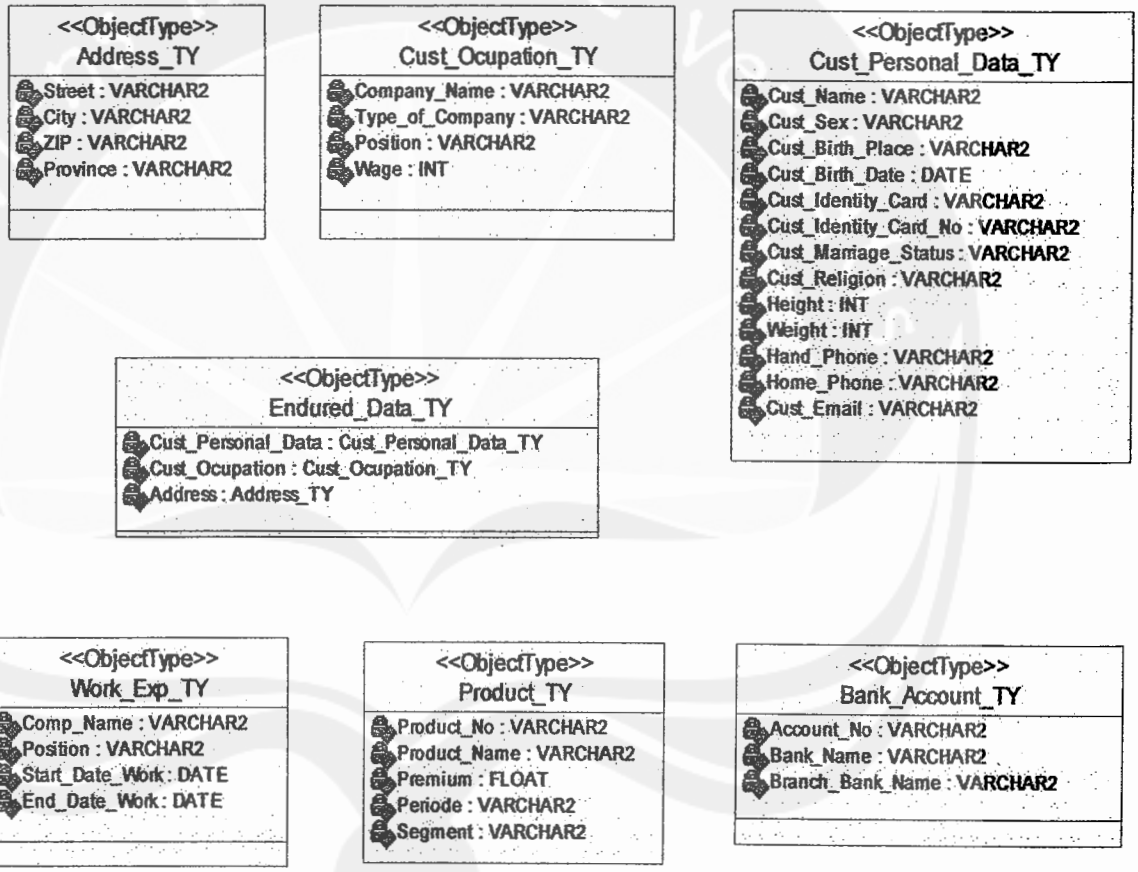
1. *User* memasukkan *keyword* untuk menentukan *report* yang ditampilkan.
2. *Control class* AllBranchReportManager melakukan *create* instance *report.*

3. *Instance report* yang baru akan dikembalikan ke *boundary class* AllBranchReportUI.
4. *Boundary class* akan mendapatkan data *branch*.
5. *Control class* akan menampilkan *report branch*.

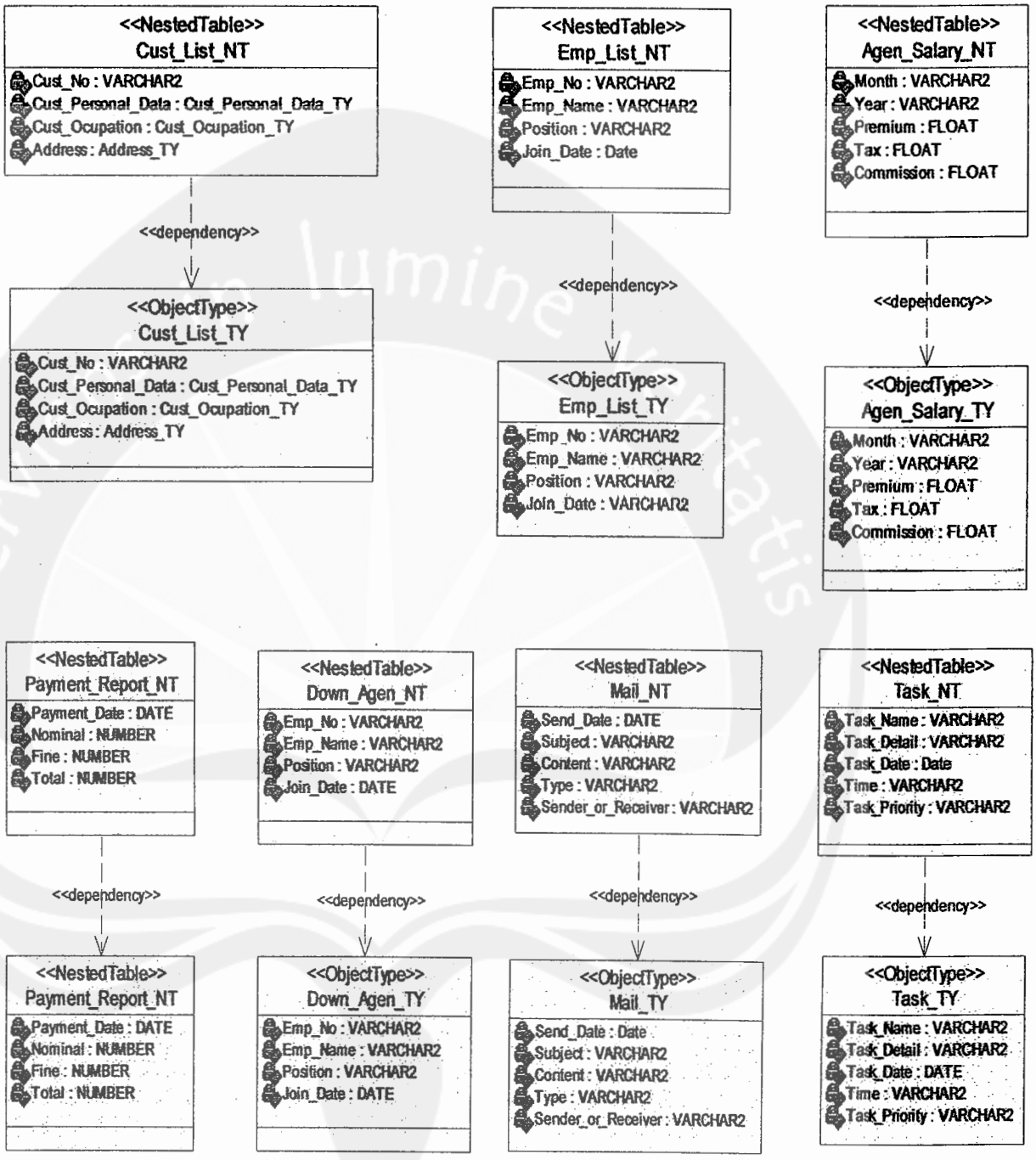
3. Deskripsi Perancangan *Persistent Data*

3.1. *Database Model*

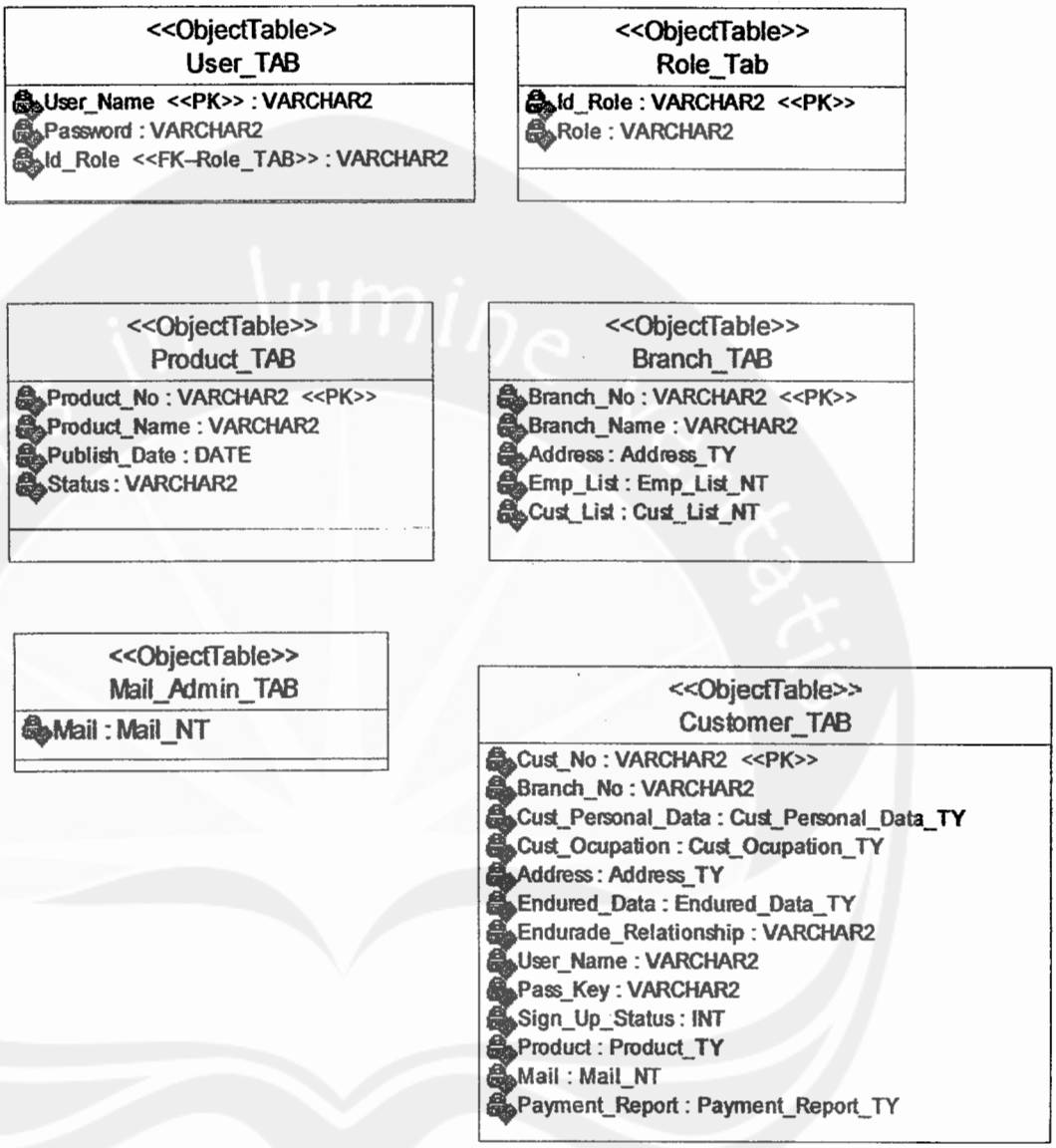
3.1.1. *Object Type*

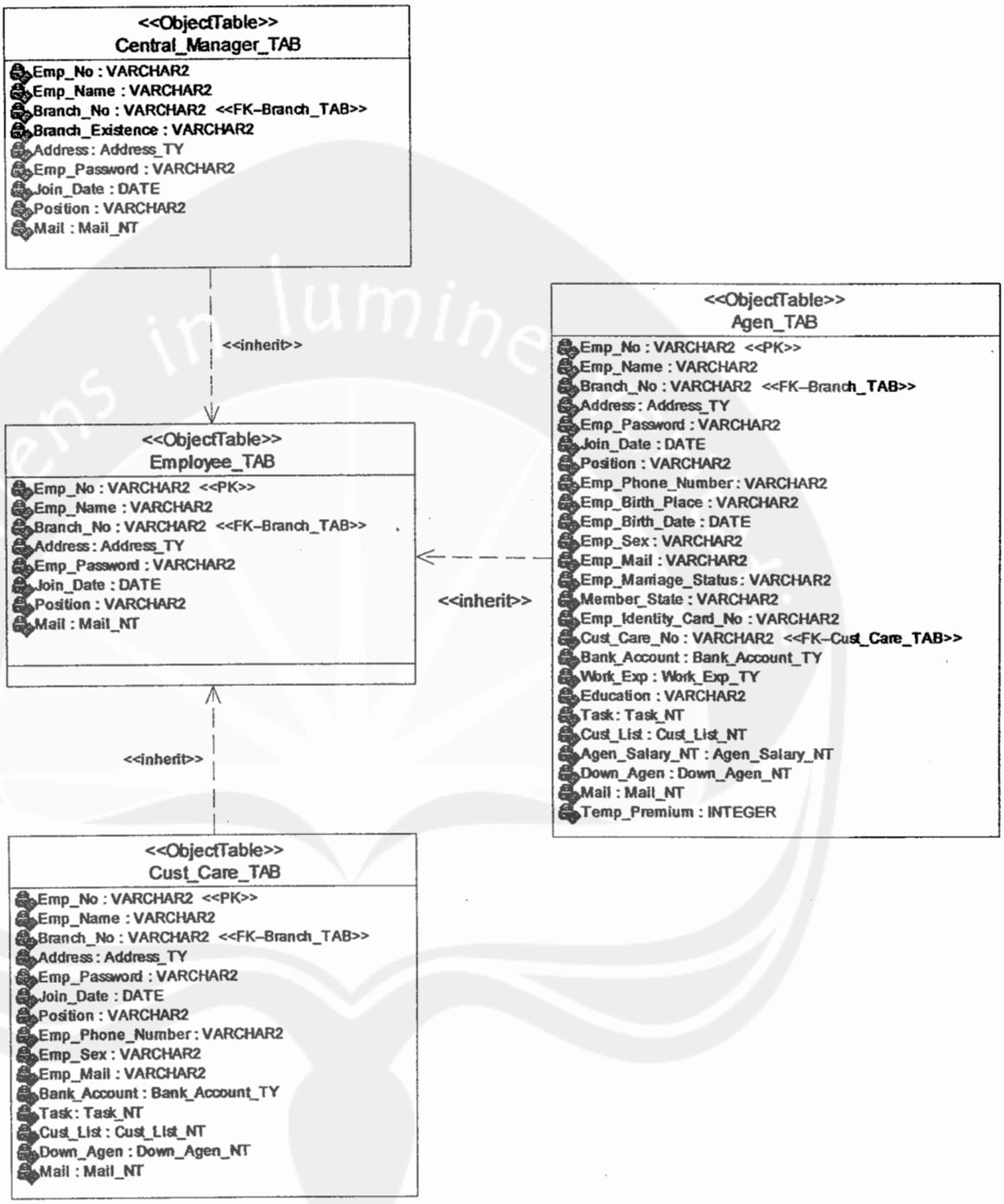


3.1.2. *Object Type* dan *Nested Table*



3.1.3. *Object Table*





3.2. *Object Type*

3.2.1. Address_TY

Object ini merepresentasikan struktur data alamat.

Field	Tipe Data	Deskripsi
Street	Varchar2(50)	Nama jalan
City	Varchar2(30)	Nama kota
ZIP	Varchar2(20)	Kodepos
Province	Varchar2(30)	Nama propinsi

Tabel 3.1 *Object Type* Address_TY

3.2.2. Cust_Ocupation_TY

Object ini merepresentasikan struktur data pekerjaan *customer*.

Field	Tipe Data	Deskripsi
Company_Name	Varchar2(30)	Nama perusahaan
Type_Of_Company	Varchar2(30)	Tipe perusahaan
Position	Varchar2(30)	Posisi/jabatan
Wage	Varchar2(30)	Gaji

Tabel 3.2 *Object Type* Cust_Ocupation_TY

3.2.3. Work_Exp_TY

Object ini merepresentasikan struktur data pengalaman bekerja pegawai/*employee*.

Field	Tipe Data	Deskripsi
Comp_Name	Varchar2(30)	Nama perusahaan
Position	Varchar2(30)	Tipe perusahaan
Start_Date	Date	Posisi/jabatan
End_Date	Date	Gaji

Tabel 3.3 *Object Type* Work_Exp_TY

3.2.4. Cust_Personal_Data_TY

Object ini merepresentasikan struktur data mengenai data pribadi *customer*.

Field	Tipe Data	Deskripsi
Cust_Name	Varchar2(50)	Nama *customer*
Cust_Sex	Varchar2(20)	Jenis kelamin
Cust_Birth_Place	Varchar2(30)	Tempat lahir
Cust_Birth_Date	Date	Tanggal lahir
Cust_Identity_Card	Varchar2(30)	Kartu identitas
Cust_Identity_Card_No	Varchar2(30)	No kartu identitas
Cust_Marriage_Status	Varchar2(30)	Status pernikahan
Cust_Religion	Varchar2(30)	Agama
Height	Varchar2(30)	Tinggi badan
Weight	Varchar2(30)	Berat badan
Hand_Phone	Varchar2(30)	No handphone
Home_Phone	Varchar2(30)	No telp rumah
Cust_Email	Varchar2(30)	Alamat email

Tabel 3.4 *Object Type* Cust_Personal_Data_TY

3.2.5. Endured_Data_TY

Object ini merepresentasikan struktur data mengenai data pribadi tertanggung.

Field	Tipe Data	Deskripsi
Cust_Personal_Data	Cust_Personal_Data_TY	Data pribadi tertanggung
Cust_Ocupation	Cust_Ocupation_TY	Pekerjaan tertanggung
Address	Address_TY	Posisi/jabatan

Tabel 3.5 *Object Type* Endured_Data_TY

3.2.6. Product_TY

Object ini merepresentasikan struktur data produk asuransi.

Field	Tipe Data	Deskripsi
Product_No	Varchar2(30)	Nomor produk
Product_Name	Varchar2(30)	Nama produk
Premium	Varchar2(30)	premi
Periode	Varchar2(30)	periode
Segment	Varchar2(30)	Segmen pasar

Tabel 3.6 *Object Type* Product_TY

3.2.7. Bank_Account_TY

Object ini merepresentasikan struktur data rekening bank.

Field	Tipe Data	Deskripsi
Account_No	Varchar2(30)	Nomor rekening
Bank_Name	Varchar2(30)	Nama bank
Branch_Bank_Name	Varchar2(30)	Nama cabang bank

Tabel 3.7 *Object Type* Bank_Account_TY

3.3. Nested Table

3.3.1. Cust_List_NT

Object ini merepresentasikan struktur data daftar *customer*.

Field	Tipe Data	Deskripsi
Cust_No	Varchar2(30)	Nomor *customer*
Cust_Personal_Data	Cust_Personal_Data_TY	Data pribadi
Cust_Ocupation	Cust_Ocupation_TY	Pekerjaan
Address	Address_TY	Alamat

Tabel 3.8 *Nested Table* Cust_List_NT

3.3.2. Emp_List_NT

Object ini merepresentasikan struktur data daftar *employee*.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(50)	Nama *employee*
Position	Varchar2(30)	Posisi
Join_Date	Date	Tanggal bergabung

Tabel 3.9 *Nested Table* Emp_List_NT

3.3.3. Agen_Salary_NT

Object ini merepresentasikan struktur data daftar gaji yang diperoleh agen.

Field	Tipe Data	Deskripsi
Month	Varchar2(30)	Bulan gaji
Year	Varchar2(30)	Tahun gaji
Premium	Number	Perolehan premi
Tax	Number	Pajak
Commission	Number	Komisi yang diterima

Tabel 3.10 *Nested Table* Agen_Salary_NT

3.3.4. Down_Agen_NT

Object ini merepresentasikan struktur data daftar agen.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(50)	Nama *employee*
Position	Varchar2(30)	Posisi
Join_Date	Date	Tanggal bergabung

Tabel 3.11 *Nested Table* Down_Agen_NT

3.3.5. Mail_NT

Object ini merepresentasikan struktur data daftar *mail* yang dimiliki *user*.

Field	Tipe Data	Deskripsi
Send_Date	Varchar2(30)	Nomor *employee*
Subject	Varchar2(50)	Nama *employee*
Content	Varchar2(30)	Posisi
Type	Date	Tanggal bergabung

Tabel 3.12 *Nested Table* Mail_NT

3.3.6. Task_NT

Object ini merepresentasikan struktur data daftar *task*(tugas) yang dimiliki *user*.

Field	Tipe Data	Deskripsi
Task_Name	Varchar2(30)	Nama tugas
Task_Detail	Varchar2(100)	Detail tugas
Task_Date	Date	Tanggal tugas
Time	Varchar2(10)	Jam tugas
Task_Proirity	Varchar2(30)	Prioritas tugas

Tabel 3.13 *Nested Table* Task_NT

3.3.7. Payment_Report_NT

Object ini merepresentasikan struktur data pembayaran premi.

Field	Tipe Data	Deskripsi
Payment_Date	Date	Tanggal pembayaran
Nominal	Number	Nominal pembayaran
Fine	Number	Denda
Total	Number	Total pembayaran

Tabel 3.14 *Nested Table* Payment_Report_NT

3.4. *Object Table*

3.4.1. User_TAB

Object ini merepresentasikan struktur data *user*/pengguna perangkat lunak.

Field	Tipe Data	Deskripsi
User_Name	Varchar2(20)	User name (PK)
Password	Varchar2(20)	Password
Id_Role	Int	Id role (FK)

Tabel 3.15 *Object Table* User_TAB

3.4.2. Role_TAB

Object ini merepresentasikan struktur data role *user*/pengguna perangkat lunak.

Field	Tipe Data	Deskripsi
Id_Role	Int	Id role (PK)
Role	Varchar2(30)	Role user

Tabel 3.16 *Object Table* Role_TAB

3.4.3. Branch_TAB

Object ini merepresentasikan struktur data *branch*/kantor cabang.

Field	Tipe Data	Deskripsi
Branch_No	Varchar2(30)	Nomor cabang (PK)
Branch_Name	Varchar2(30)	Nama cabang
Address	Address_TY	Alamat cabang
Emp_List	Emp_List_TY	Daftar *employee*
Cust_List	Cust_List_TY	Daftar *customer*

Tabel 3.17 *Object Table* Branch_TAB

3.4.4. Product_TAB

Object ini merepresentasikan struktur data produk asuransi.

Field	Tipe Data	Deskripsi
Product_No	Varchar2(30)	Nomor produk (PK)
Product_Name	Varchar2(30)	Nama produk
Publish_Date	Date	Tanggal diluncurkan
Status	Varchar2(30)	Status produk

Tabel 3.18 *Object Table* Product_TAB

3.4.5. Mail_Admin_TAB

Object ini merepresentasikan struktur data daftar *mail* yang dimiliki *admin*.

Field	Tipe Data	Deskripsi
Mail	Mail_NT	Daftar *mail*

Tabel 3.19 *Object Table* Mail_Admin_TAB

3.4.6. Employee_TAB

Object ini merepresentasikan struktur data *employee*/pegawai.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(30)	Nama *employee*
Branch_No	Varchar2(30)	Nomor cabang
Address	Address_TY	Alamat *employee*
Emp_Password	Varchar2(30)	*Password*
Join_Date	Date	Tanggal bergabung
Position	Varchar2(30)	Posisi
Mail	Mail_NT	Daftar *mail employee*

Tabel 3.20 *Object Table* Employee_TAB

3.4.7. Customer_TAB

Object ini merepresentasikan struktur data produk asuransi.

Field	Tipe Data	Deskripsi
Cust_No	Varchar2(30)	Nomor cusomer (PK)
Branch_No	Varchar2(30)	Nomor cabang (FK)
Cust_Personal_Data	Cust_Personal_Data_TY	Data pribadi *customer*
Cust_Ocupation	Cust_Ocupation_TY	Perkerjaan *customer*
Address	Address_TY	Alamat *customer*
Endured_Data	Endured_Data_TY	Data tertanggung
Endured_Relationship	Varchar2(30)	Hubungan dgn tertanggung
User_Name	Varchar2(30)	*User name customer*
Pass_Key	Varchar2(30)	*Password customer*
Sign_Up_Status	Char(1)	Status pendaftaran
Product	Product_TY	Produk asuransi
Mail	Mail_NT	Daftar *mail customer*
Payment_Report	Payment_Report_NT	Laporan pembayaran

Tabel 3.21 Object Table Customer_TAB

3.4.8. Central_Manager_TAB

Object ini merepresentasikan struktur data *central manager*.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(30)	Nama *employee*
Branch_No	Varchar2(30)	Nomor cabang
Branch_Existence	Varchar2(30)	Lokasi Keberadaan *central manager*
Address	Address_TY	Alamat *employee*
Emp_Password	Varchar2(30)	Password

Join_Date	Date	Tanggal bergabung
Position	Varchar2(30)	Posisi
Mail	Mail_NT	Daftar *mail employee*

Tabel 3.22 *Object Table* Central_Manager_TAB

3.4.9. Agen_TAB

Object ini merepresentasikan struktur data agen.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(30)	Nama *employee*
Branch_No	Varchar2(30)	Nomor cabang
Address	Address_TY	Alamat *employee*
Emp_Password	Varchar2(30)	Password
Join_Date	Date	Tanggal bergabung
Position	Varchar2(30)	Posisi
Emp_Phone_Number	Varchar2(30)	Nomor telp
Emp_Birth_Place	Varchar2(30)	Tempat lahir
Emp_Birth_Date	Date	Tanggal lahir
Emp_Sex	Varchar2(20)	Jenis kelamin
Emp_Email	Varchar2(30)	Alamat email
Emp_Marriage_Status	Varchar2(20)	Status pernikahan
Member_State	Varchar2(20)	Warga negara
Identity_Card_No	Varchar2(30)	Nomor kartu identitas
Bank_Account	Bank_Account_TY	Data rekening bank
Work_Exp	Work_Exp_TY	Pengalaman kerja
Education	Education_TY	Pendidikan terakhir
Task	Task_NT	Daftar *task*/jadwal
Cust_List	Cust_List_NT	Daftar customer yang didapat
Agen_Salary	Agen_Salary_NT	Daftar gaji

Down_Agen	Down_Agen_NT	Daftar agen bawahan
Mail	Mail_NT	Daftar *mail employee*
Temp_Premium	Number	Total premi yang didapat per bulan

Tabel 3.23 *Object Table* Agen_TAB

3.4.10. Cust_Care_TAB

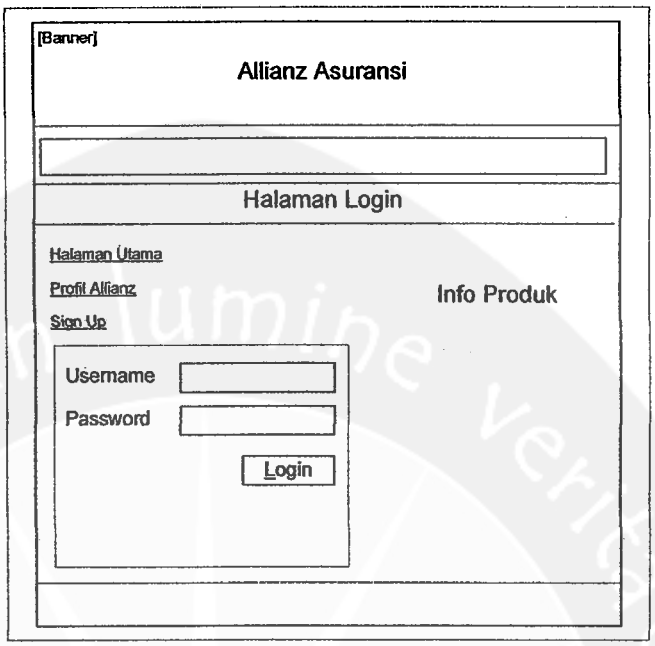
Object ini merepresentasikan struktur data *customer care*.

Field	Tipe Data	Deskripsi
Emp_No	Varchar2(30)	Nomor *employee*
Emp_Name	Varchar2(30)	Nama *employee*
Branch_No	Varchar2(30)	Nomor cabang
Address	Address_TY	Alamat *employee*
Emp_Password	Varchar2(30)	Password
Join_Date	Date	Tanggal bergabung
Position	Varchar2(30)	Posisi
Emp_Phone_Number	Varchar2(30)	Nomor telp
Emp_Sex	Varchar2(20)	Jenis kelamin
Emp_Email	Varchar2(30)	Alamat email
Bank_Account	Bank_Account_TY	Data rekening bank
Task	Task_NT	Daftar *task*/jadwal
Cust_List	Cust_List_NT	Daftar customer yang didapat
Down_Agen	Down_Agen_NT	Daftar agen bawahan
Mail	Mail_NT	Daftar *mail employee*

Tabel 3.24 *Object Table* Cust_Care_TAB

4. Deskripsi Perancangan Antarmuka

4.1. *Use Case : Login*



Gambar 4.1 Rancangan Halaman Login

Deskripsi

- Halaman ini digunakan oleh *user* untuk *login*. Hak akses terhadap operasi yang dapat dilakukan *user*, tergantung dari *user role*-nya.
- Pada halaman ini, terdapat sebuah tombol untuk meng-konfirmasi data *login* yang *user* masukkan.

Event

- *Login*

```
onClick_Login()
{
    SQL Statement :
    SELECT * FROM User_TAB
    WHERE User_Name = '[txtUserName]'
    AND Password = '[txtPassword]'
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan *user name* dan *password* pada masing-masing *textbox* yang tersedia.
2. *User* menekan tombol *Login*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *login* yang *user* masukkan, apabila *valid*, maka akan terbuka halaman sesuai dengan jenis akses (*role*) *user*.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

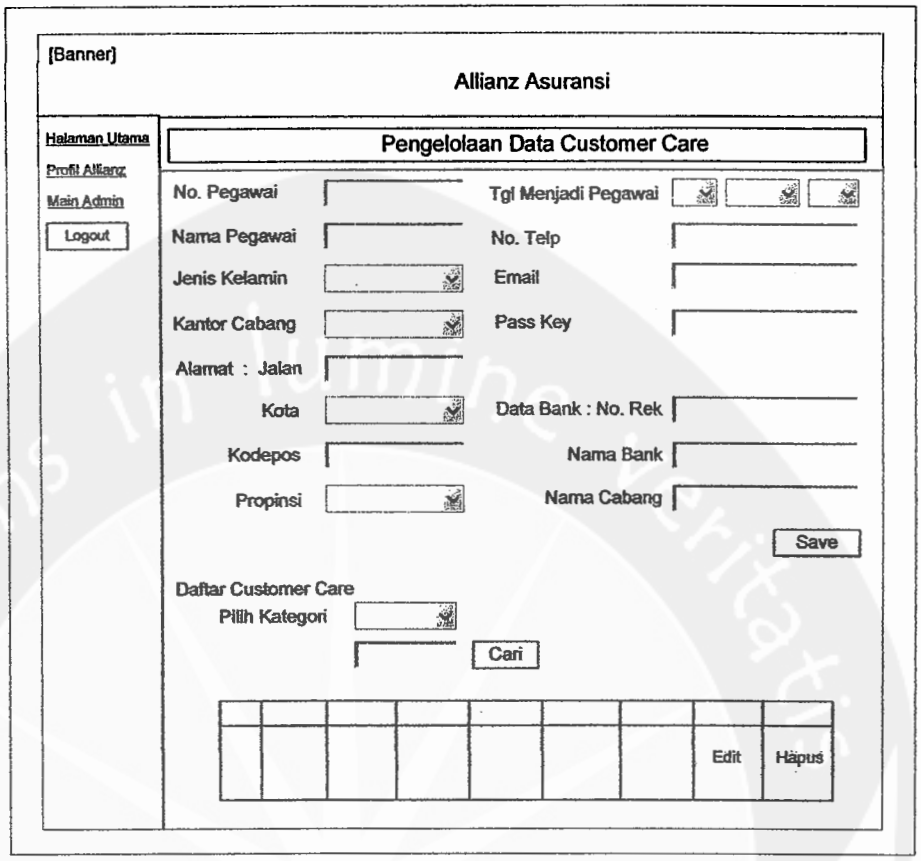
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Sign Up*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman *sign up* dan menyediakan fasilitas bagi *user* untuk melakukan *sign up*.

4.2. *Use Case Manage Customer Care - Add Customer Care*



Gambar 4.2 Rancangan Halaman *Add Customer Care*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *customer care*. Penambahan data yang dilakukan akan mempengaruhi tabel Cust_Care_TAB, User_TAB, dan Branch_TAB.
- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.

- cari → tombol untuk melakukan pencarian data *customer care*.
- edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
- hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    address = ('[txtJalan]','[ddlKota]','[txtKodepos]',
          '[ddlPropinsi]');
    bank_account = ('[txtNoRek]','[txtNamaBank]',
              '[txtNamaCabang]');
    branch_no = getBranchNo('[txtBranchName]');
    join_date = ('[ddlTgl]','[ddlBln]','[ddlThn]');

    SQL Statement :
    INSERT INTO Cust_Care_TAB (Emp_No, Emp_Name, Emp_Sex,
    Branch_No, Address, Pass_Key, Join_Date, Phone_No,
    Sex, Email)
    VALUES ('[txtEmpNo]','[txtEmpName]','[ddlSex]',
    [branch_no],[address],'[txtPassKey]',[join_date],
    '[txtPhoneNo]','[ddlSex]','[txtEmail]'
    [bank_account]);

    INSERT INTO User_TAB (User_Name, Password, Id_Role)
    VALUES ('[txtEmpNo]','[txtPassKey]','3');

    INSERT INTO TABLE (SELECT Emp_List FROM Branch_TAB
    WHERE Branch_No = [branch_no])
    VALUES (Emp_List_TY('[txtEmpNo]','[txtEmpName]',
    'Customer Care',[join_date]));
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *customer care* pada masing-masing *textbox* yang tersedia.

2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *customer care* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

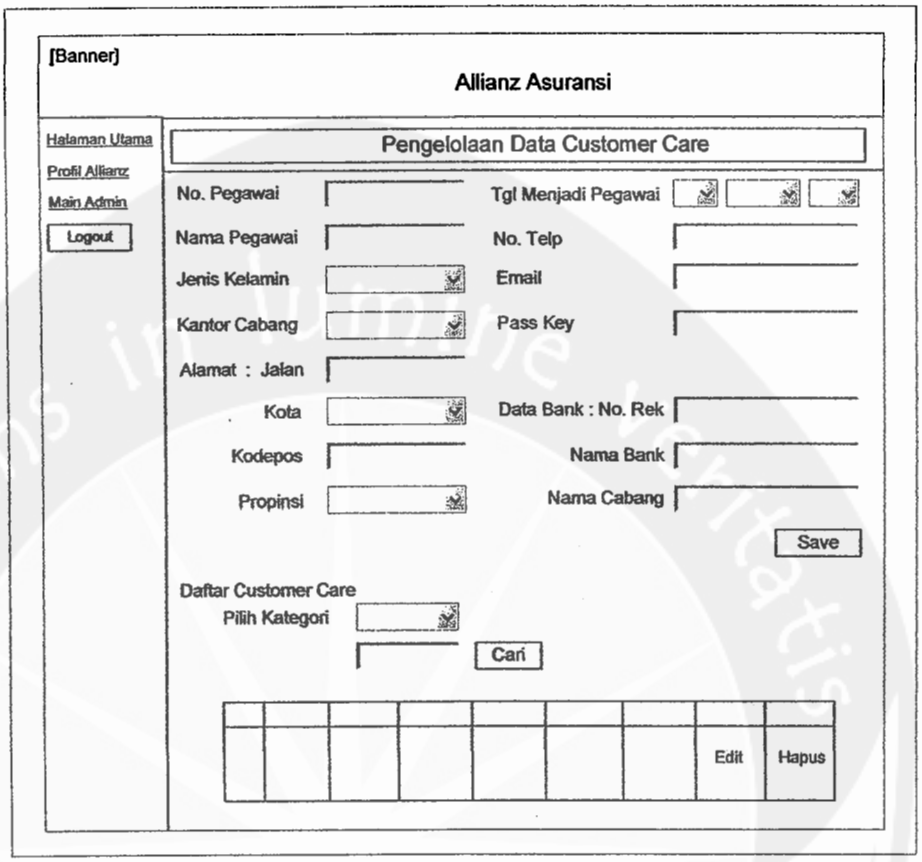
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.3. *Use Case Manage Customer Care - Edit Customer Care*



Gambar 4.3 Rancangan Halaman *Edit Customer Care*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *customer care*. Pengeditan data tersebut akan mempengaruhi tabel Cust_Care_TAB dan User_TAB.
- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.

- cari → tombol untuk melakukan pencarian data *customer care*.
- edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
- hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    address = ('[txtJalan]','[ddlKota]','[txtKodepos]',
          '[ddlPropinsi]');
    bank_account = ('[txtNoRek]','[txtNamaBank]',
              '[txtNamaCabang]');
    branch_no = getBranchNo('[txtBranchName]');
    join_date = ('[ddlTgl]','[ddlBln]','[ddlThn]');

    SQL Statement :
    UPDATE Cust_Care_TAB
    SET Emp_No = '[txtEmpNo]',Emp_Name = '[txtEmpName]',
    Branch_No = [branch_no],Address = [address],
    Pass_Key = '[txtPassKey]',Join_Date = [join_date],
    Phone_No = '[txtPhoneNo]',Emp_Sex = '[txtSex]',
    Emp_Email = '[txtEmail]'
    WHERE Emp_No = '[txtEmpNo]';

    UPDATE User_TAB
    SET Password = '[txtPassKey]'
    WHERE User_Name = '[txtUserName]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.

2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol edit pada *datagrid customer care* yang bersangkutan, maka data tersebut akan tertampil pada *textbox* sesuai dengan *field*-nya.
4. *User* melakukan perubahan terhadap data *customer care*.
5. *User* menekan tombol *Save*.
6. *Perangkat* lunak akan melakukan pengecekan terhadap data *customer care* yang *user* masukkan, apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Cust_Care_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.

2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

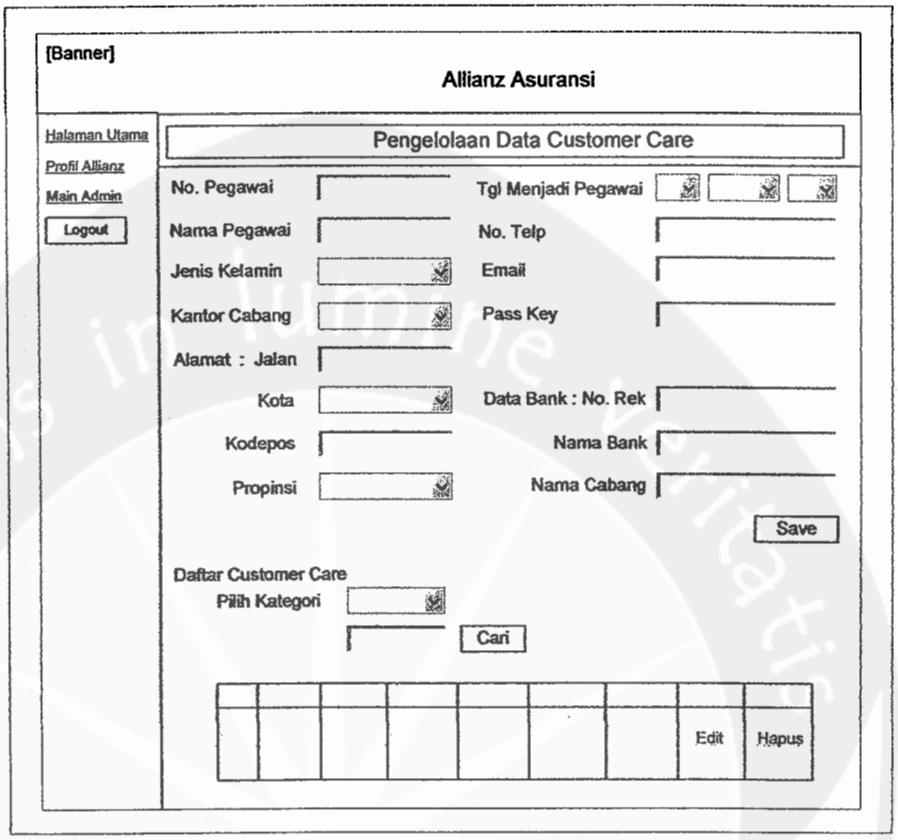
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin.*
  2. Perangkat lunak akan membuka utama administrator.

4.4. *Use Case Manage Customer Care - Delete Customer Care*



Gambar 4.4 Rancangan Halaman *Delete Customer Care*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penghapusan data *customer care*. Penghapusan data tersebut akan mempengaruhi tabel Cust_Care_TAB, User_TAB, dan Branch_TAB.
- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.

- cari → tombol untuk melakukan pencarian data *customer care*.
- edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
- hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesuai dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Delete*

```
onClick_Delete()
{
    branch_no = getBranchNo ('[txtEmpNo]')
    SQL Statement :
    UPDATE Agen_TAB
    SET Cust_Care_No = null
    WHERE Branch_No = [branch_no];

    DELETE FROM Cust_Care_TAB
    WHERE Emp_No = '[txtEmpNo]';

    DELETE FROM TABLE (SELECT Emp_List FROM Branch_TAB
    WHERE Branch_No = [branch_no])
    WHERE Emp_No = '[txtEmpNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol delete pada baris datagrid sesuai dengan data customer care yang akan dihapus.

4. Perangkat lunak akan melakukan penghapusan data *customer care* dari *database* sesuai pilihan *user*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Cust_Care_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. User klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

Urutan aksi yang terjadi :

1. *User* melakukan klik *link* halaman utama.
2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

Urutan aksi yang terjadi :

1. *User* melakukan klik *link* profil Allianz.

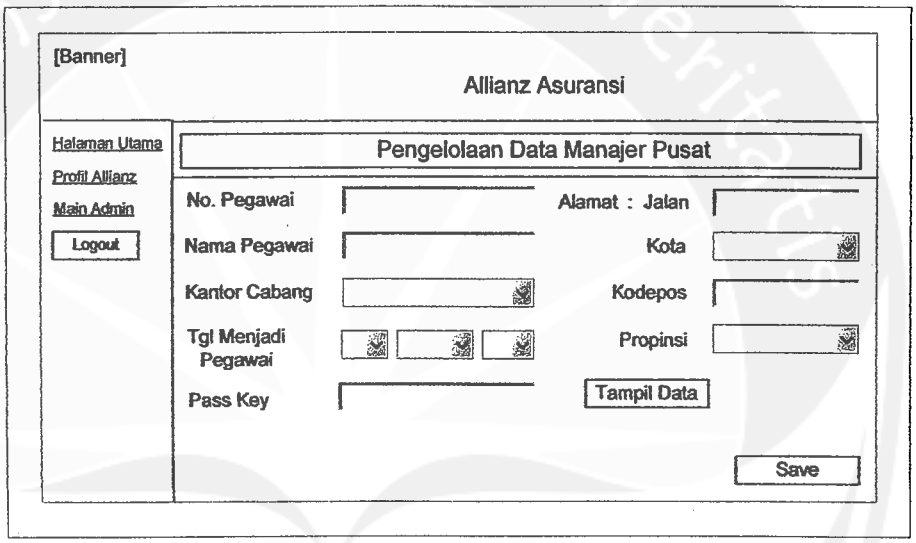
2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.5. *Use Case Manage Central Manager - Add Central Manager*



Gambar 4.5 Rancangan Halaman *Add Central Manager*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *central manager*. Penambahan data tersebut akan mempengaruhi tabel Central_Manager_TAB dan User_TAB.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:

- *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
- Tampil data → tombol yang digunakan untuk menampilkan data mengenai manajer pusat yang saat ini menjabat.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    branch_no = getBranchNo('[ddlCabang]');
    address = ('[txtJalan]','[ddlKota]','[txtKodepos]',
        '[ddlPropinsi]');
    join_date = ('[ddlTgl]','[ddlBln]','[ddlThn]');

    SQL Statement :
    INSERT INTO Central_Manager_TAB (Emp_No,Emp_Name,
    Branch_No,Address,Branch_Existence,Pass_Key,
    Join_Date,Position)
    VALUES ('[txtEmpNo]','[txtEmpName]',[branch_no],
    [address],'[ddlCabang]','[txtPassKey]',[join_date],
    'Central Manager');

    INSERT INTO User_TAB
    VALUES ('[txtEmpNo]','[txtPassKey]',
    'Central Manager');
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *central manager* pada masing-masing *textbox* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *central manager* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

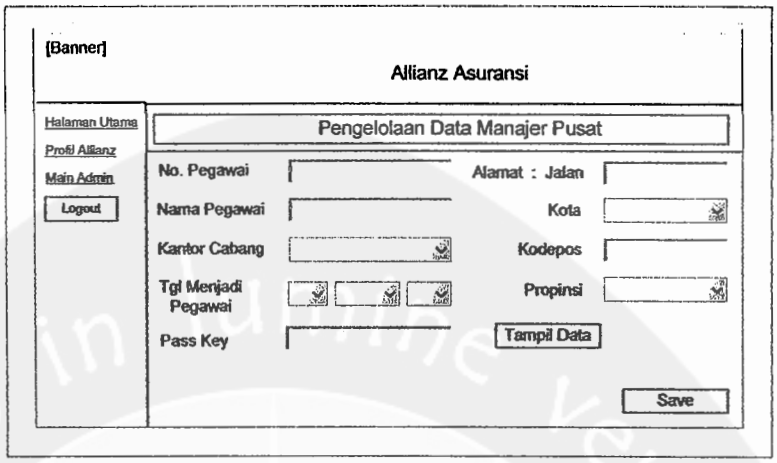
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.6. *Use Case Manage Central Manager - Edit Central Manager*



Gambar 4.6 Rancangan Halaman *Edit Central Manager*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *central manager*. Pengeditan data tersebut akan mempengaruhi tabel Central_Manager_TAB dan User_TAB.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data customer care.
  - Tampil data → tombol yang digunakan untuk menampilkan data mengenai manajer pusat yang saat ini menjabat.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
```

```
branch_no = getBranchNo('[ddlCabang]');
address = ('[txtJalan]',' [ddlKota]',' [txtKodepos]',
      ' [ddlPropinsi]');
join_date = ('[ddlTgl]','[ddlBln]','[ddlThn]');

SQL Statement :
DELETE FROM User_TAB
WHERE User_Name = '[txtEmpNo]';

DELETE FROM Central_Manager_TAB
WHERE Emp_No = '[txtEmpNo]';

INSERT INTO Central_Manager_TAB (Emp_No,Emp_Name,
Branch_No,Address,Branch_Existence,Pass_Key,
Join_Date,Position)
VALUES ('[txtEmpNo]','[txtEmpName]',[branch_no],
[address],'[ddlCabang]','[txtPassKey]',[join_date],
'Central Manager');


INSERT INTO User_TAB
VALUES ('[txtEmpNo]','[txtPassKey]',
'Central Manager');
}
```

Urutan aksi yang terjadi :

1. *User* klik tombol tampil data.
2. *User* memasukkan semua data *central manager* (mengedit) pada masing-masing *textbox* yang tersedia, lalu tekan tombol *save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *central manager* yang *user* masukkan, apabila *valid*, maka data tersebut akan diolah ke dalam *database*.

- Tampil Data

```
onClick_Tampil_Data()
{
    SQL Statement :
    SELECT * FROM Central_Manager_TAB;
}
```

Urutan aksi yang terjadi :

1. *User* klik tombol tampil data.

2. Perangkat lunak akan menampilkan data *central manager* ke *textbox* yang ada.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

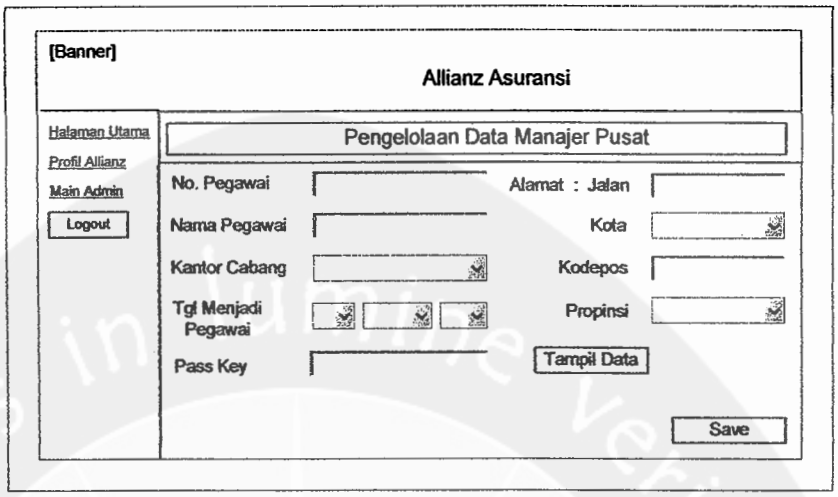
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.7. *Use Case Manage Central Manager – Display Central Manager*



Gambar 4.7 Rancangan Halaman *Display Central Manager*

Deskripsi

- Halaman ini digunakan oleh *user* untuk menampilkan data yang dimiliki oleh manajer pusat.

- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - Tampil data → tombol yang digunakan untuk menampilkan data mengenai manajer pusat yang saat ini menjabat.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- Tampil Data

```
onClick_Tampil_Data()
{
    SQL Statement :
    SELECT * FROM Central_Manager_TAB;
}
```

Urutan aksi yang terjadi :

1. *User* klik tombol tampil data.
2. Perangkat lunak akan menampilkan data *central manager* ke *textbox* yang ada.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

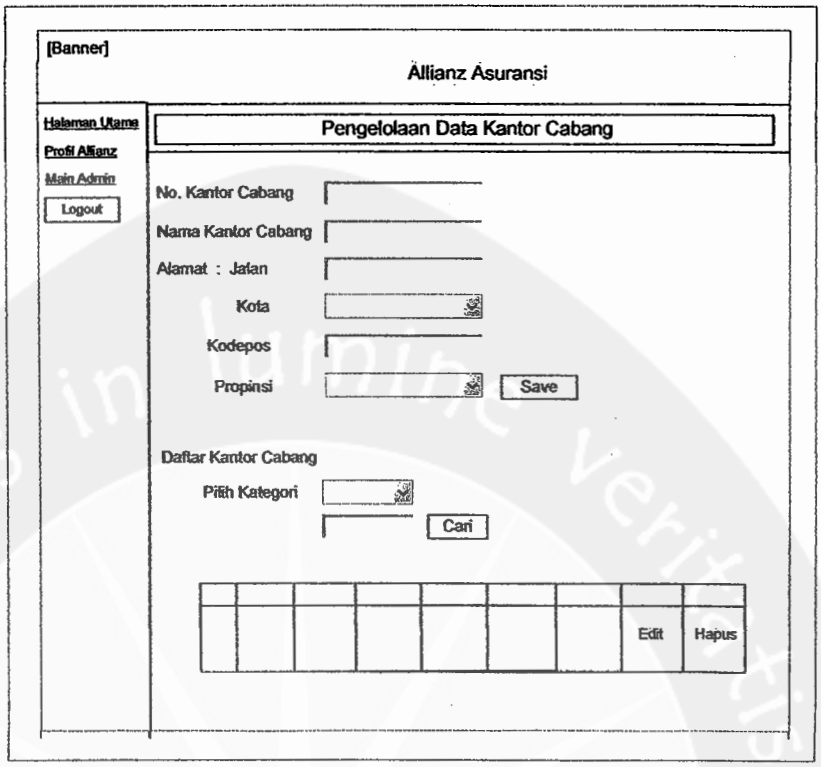
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.8. *Use Case Manage Branch – Add Branch*



Gambar 4.8 Rancangan Halaman *Add Branch*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *branch*. Penambahan data tersebut akan mempengaruhi tabel Branch_TAB.
- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *branch*.

- cari → tombol untuk melakukan pencarian data *branch*.
- edit → tombol untuk memulai proses *editing* terhadap data *branch*.
- hapus → tombol untuk melakukan penghapusan terhadap data *branch* sesuai dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    address = ('[txtJalan]','[ddlKota]','[txtKodepos]',
        '[ddlPropinsi]');

    SQL Statement :
    INSERT INTO Branch_TAB (Branch_No,Branh_Name,Address)
    VALUES ('[txtBranchNo]','[txtBranchName]',
    '[address]')
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *branch* pada masing-masing *textbox* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *branch* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.

2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

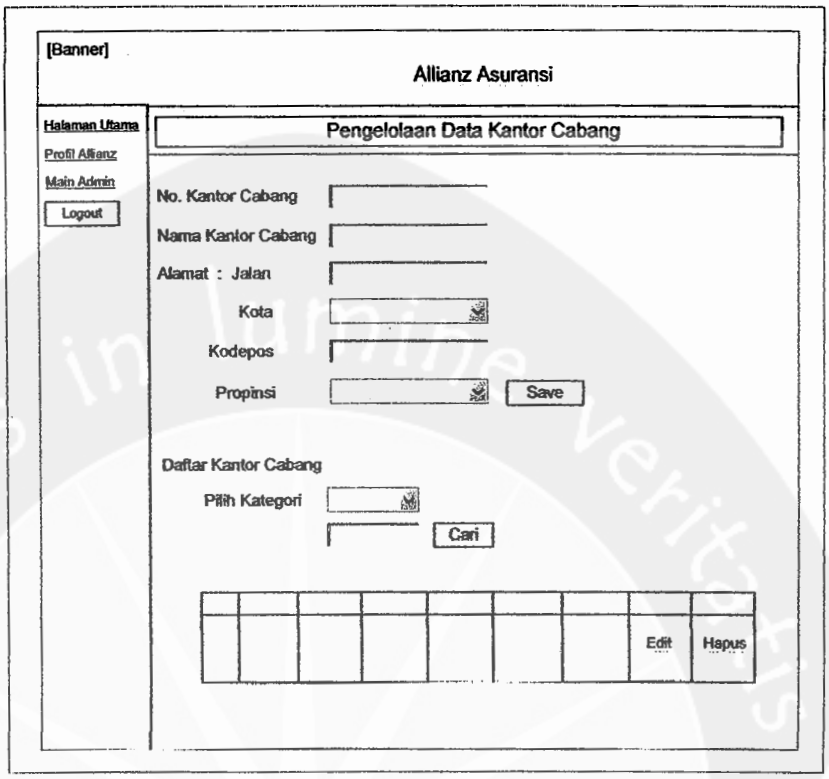
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.9. *Use Case Manage Branch – Edit Branch*



Gambar 4.9 Rancangan Halaman *Edit Branch*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *branch*. Pengeditan data tersebut akan mempengaruhi tabel Branch_TAB.
- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *branch*.
  - cari → tombol untuk melakukan pencarian data *branch*.
  - edit → tombol untuk memulai proses *editing* terhadap data *branch*.

- hapus → tombol untuk melakukan penghapusan terhadap data *branch* sesusi dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    address = ('[txtJalan]',' [ddlKota]',' [txtKodepos]',
          ' [ddlPropinsi]');

    SQL Statement :
    UPDATE Branch_TAB
    SET Branch_Name = '[txtBranchName]',
    Address = '[address]'
    WHERE Branch_No = '[txtBranchNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol edit pada *datagrid branch* yang bersangkutan, maka data tersebut akan tertampil pada *textbox* sesuai dengan *field*-nya.
4. *User* melakukan perubahan terhadap data *branch*.
5. *User* menekan tombol *Save*.
6. *Perangkat* lunak akan melakukan pengecekan terhadap data *branch* yang *user* masukkan, apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Branch_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

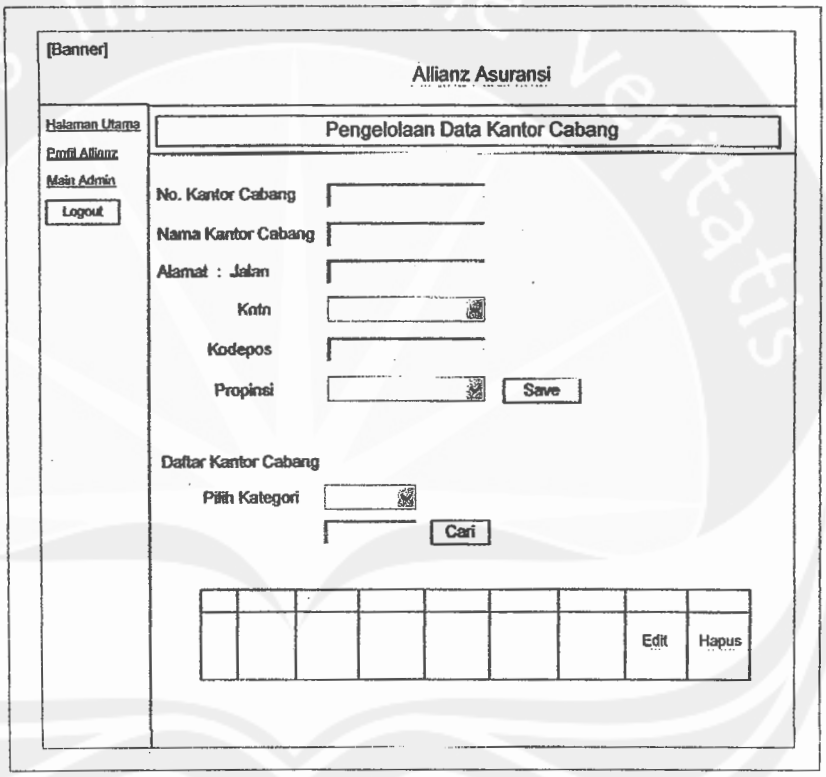
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.10. *Use Case Manage Branch - Delete Branch*



Gambar 4.10 Rancangan Halaman *Delete Branch*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penghapusan data *branch*. Penghapusan data tersebut akan mempengaruhi tabel Branch_TAB.

- Pada halaman ini, terdapat 5 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Delete*

```
onClick_Delete()
{    SQL Statement :
     DELETE FROM Branch_TAB
     WHERE Branch_No = '[txtBranchNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol hapus pada baris datagrid sesuai dengan data *branch* yang akan dihapus.

4. Perangkat lunak akan melakukan penghapusan data *branch* dari *database* sesuai pilihan *user*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Branch_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

Urutan aksi yang terjadi :

1. *User* melakukan klik *link* halaman utama.
2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

Urutan aksi yang terjadi :

1. *User* melakukan klik *link* profil Allianz.

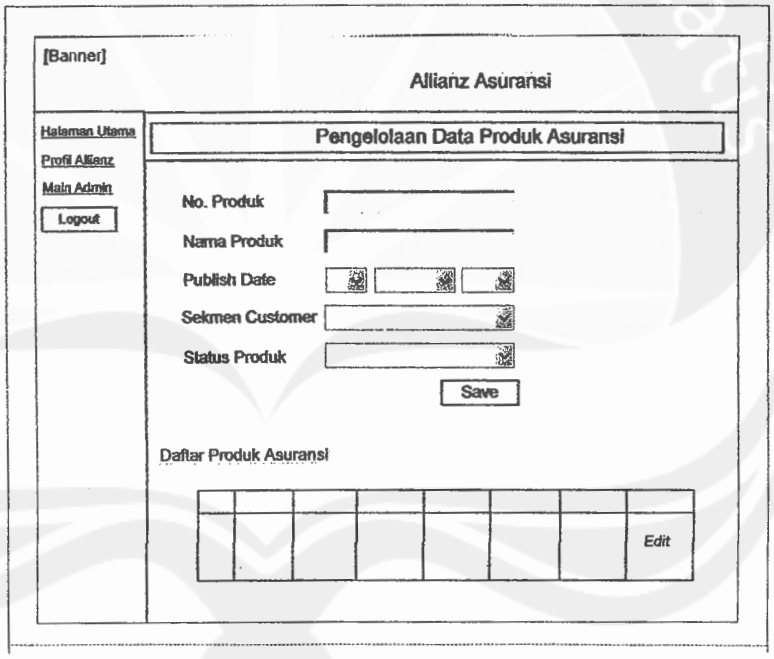
2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.11. *Use Case Manage Product - Add Product*



Gambar 4.11 Rancangan Halaman *Add Product*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *product*. Penambahan data tersebut akan mempengaruhi tabel Product_TAB.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *branch*.
  - *edit* → tombol untuk melakukan pengeditan terhadap data *product* sesuai dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    publish_date = ('[ddlTgl]','[ddlBln]','[ddlThn]');

    SQL Statement :
    INSERT INTO Product_TAB
    VALUES ('[txtProductNo]','[txtProductName]',
    '[publish_date]','[ddlSegment]', '[ddlStatus]')
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *product* pada masing-masing *textbox* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *product* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

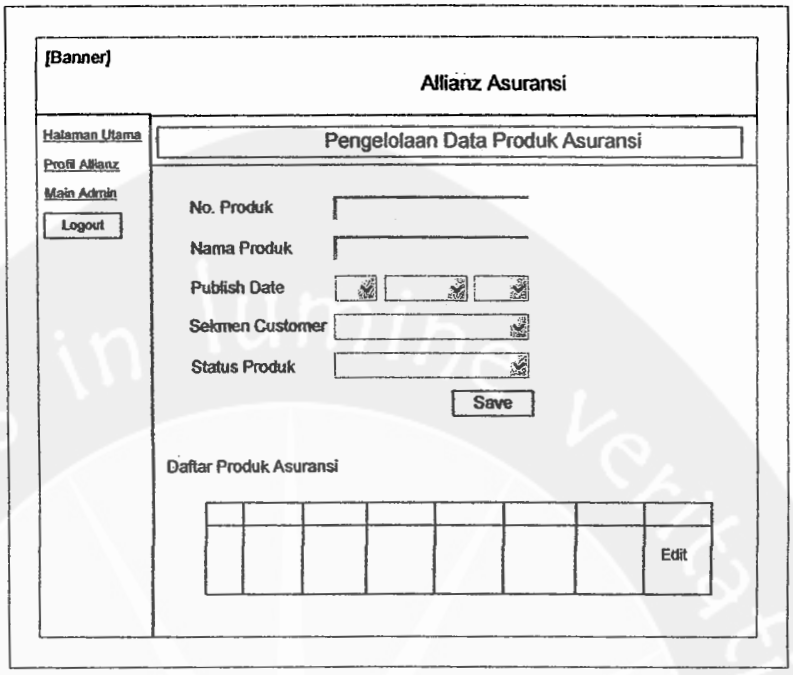
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin*.
  2. Perangkat lunak akan membuka utama administrator.

4.12. *Use Case Manage Product - Edit Product*



Gambar 4.12 Rancangan Halaman *Edit Product*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *product*. Pengeditan data tersebut akan mempengaruhi tabel Product_TAB.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *branch*.

- *edit* → tombol untuk melakukan pengeditan terhadap data *product* sesuai dengan pilihan *user*.
- *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    address = ('[txtJalan]',' [ddlKota]',' [txtKodepos]',
          ' [ddlPropinsi]');

    SQL Statement :
    UPDATE Product_TAB
    SET Status = '[ddlStatus]',
    Segment = '[ddlSegment]'
    WHERE Product_No = '[txtProductNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol edit pada *datagrid product* yang bersangkutan, maka data tersebut akan tertampil pada *textbox* sesuai dengan *field*-nya.
4. *User* melakukan perubahan terhadap data *product*.
5. *User* menekan tombol *Save*.
6. *Perangkat* lunak akan melakukan pengecekan terhadap data *product* yang *user* masukkan,

apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

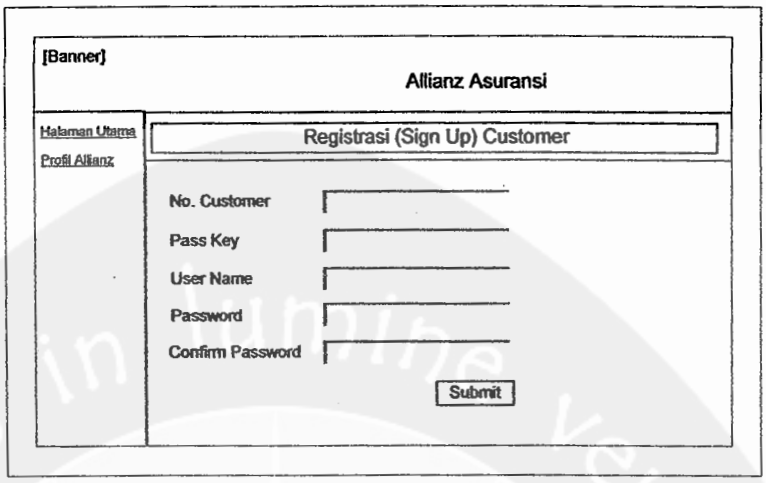
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.13. *Use Case Customer Check*



Gambar 4.13 Rancangan Halaman *Customer Check*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan registrasi agar dapat menggunakan layanan-layanan CRM sesuai dengan *account*-nya. Registrasi ini akan mempengaruhi 2 tabel, yaitu User_TAB dan Customer_TAB.
- Pada halaman ini, terdapat sebuah tombol, yaitu:
  - *submit* → tombol untuk melakukan eksekusi registrasi.

Event

- *Submit*

```
onClick_Submit()
{
    SQL Statement :
    UPDATE Customer_TAB
    SET User_Name = '[txtUserName]',
    Pass_Key = '[txtPassword]'
    WHERE Cust_No = '[txtCustNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memasukkkan data yang diperlukan untuk registrasi.
2. *User* klik tombol submit.
3. Perangkat lunak akan melakukan pengecekan terhadap data registrasi, apabila valid, maka database akan diupdate.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

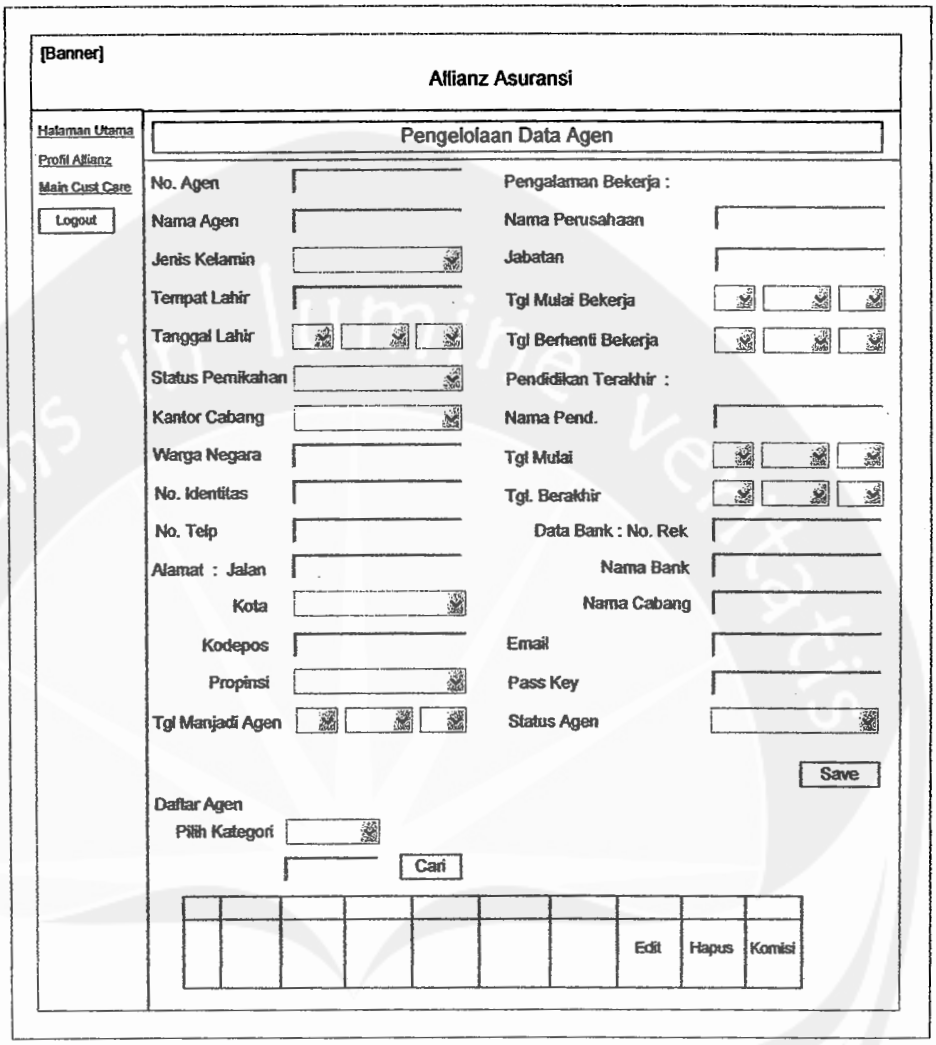
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz

4.14. *Use Case Manage* Agen – *Add* Agen



Gambar 4.14 Rancangan Halaman *Add* Agen

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data agen. Penambahan data tersebut akan mempengaruhi 4 tabel yaitu Agen_TAB, User_TAB, *nested table* Down_Agen, dan *nested table* Emp_List.

- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - komisi → tombol untuk menuju ke halaman laporan mengenai komisi yang agen terima.

Event

- *Save*

```
onClick_Save()
{
    join_date = ('[ddlTglJoin]','[ddlBlnJoin]',
                '[ddlThnJoin]');
    birth_date = ('[ddlTglLahir]','[ddlBlnLahir]',
                '[ddlThnLahir]');
    start_work_date = ('[ddlTglStartWork]',
                '[ddlBlnStartWork]',
                '[ddlThnStartWork]');
    end_word_date = ('[ddlTglEndWork]','[ddlBlnEndWork]',
                '[ddlThnEndWork]');
    start_work_edu = ('[ddlTglStartEdu]',
                '[ddlBlnStartEdu]',
                '[ddlThnStartEdu]');
    end_word_edu = ('[ddlTglEndEdu]','[ddlBlnEndEdu]',
                '[ddlThnEndEdu]');
    address = ('[txtJalan]',' [ddlKota]',' [txtKodepos]',
                ' [ddlPropinsi]');
    branch_no = getBranchNo('[ddlCabang]') ;
    cust_care_no = getCustNo();
```

```
bank_account = ('[txtNoRek]','[txtNamaBank]',
               '[txtNamaCabang]');
work_exp = ('[txtNamaPerusahaan]','[txtJabatan]',
           [start_work_date],[end_work_date]);
education = ('[txtNamaPend]','[ddlBlnStartWork]',
           '[ddlThnStartWork]');

SQL Statement :
INSERT INTO Agen_TAB (Emp_No,Emp_Name,Branch_No,
Address,Emp_Password,Join_Date,Position,Status,
Emp_Phone_No,Emp_Birth_Place,Emp_Birth_Date,Emp_Sex,
Emp_Email,Emp_Marriage_Status,Member_State,
Identity_Card_No,Cust_Care_No,Bank_Account,Woek_Exp,
Education)
VALUES ('[txtEmpNo]','[txtEmpName]',[branch_no]
[address],'[empPassKey]',[join_date],'Agen',
'[ddlStatus]','[txtPhoneNo]','[txtBirthPlace]',
[birth_date],'[ddlSex]','[txtEmail]',
'[ddlStatusNikah]','[txtWargaNegara]',
'[txtEmpCardNo]',[cust_care_no],[bank_account],
[work_exp],[education]);

INSERT INTO User_TAB
VALUES ('[txtEmpNo]','[txtPassKey]','Agen');

INSERT INTO TABLE (SELECT Down_Agen
FROM Cust_Care_TAB
WHERE Emp_No = current_user.empNo)
VALUES (Down_Agen_TY('[txtEmpNo]',
'[txtEmpName]','Agen',[join_date]));

INSERT INTO TABLE (SELECT Emp_List
FROM Branch_TAB
WHERE Branch_No = current_user.branchNo)
VALUES (Down_Agen_TY('[txtEmpNo]',
'[txtEmpName]','Agen',[join_date]));
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data agen pada masing-masing *textbox* dan *combo box* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data agen yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

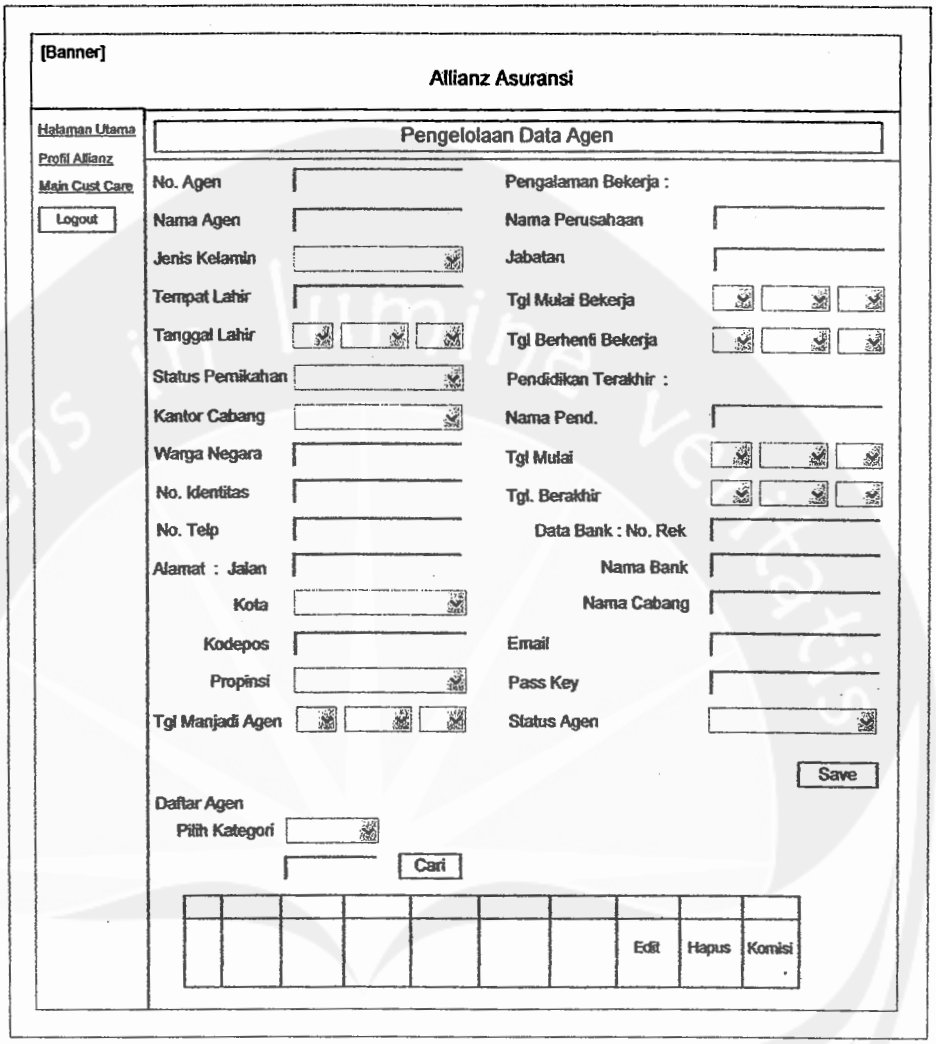
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Customer Care*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main customer care.*
  2. Perangkat lunak akan membuka utama *customer care.*

4.15. *Use Case Manage* Agen – *Edit* Agen



Gambar 4.15 Rancangan Halaman *Edit* Agen

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data agen. Pengeditan data tersebut akan mempengaruhi 4 tabel, yaitu Agen_TAB, *nested table* Emp_List, *nested table* Down_Agen, dan User_TAB.

- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi *penambahan* ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - komisi → tombol untuk menuju ke halaman laporan mengenai komisi yang agen terima.

Event

- *Save*

```
onClick_Save()
{
    join_date = ('[ddlTglJoin]','[ddlBlnJoin]',
                '[ddlThnJoin]');
    birth_date = ('[ddlTglLahir]','[ddlBlnLahir]',
                '[ddlThnLahir]');
    start_work_date = ('[ddlTglStartWork]',
                '[ddlBlnStartWork]',
                '[ddlThnStartWork]');
    end_word_date = ('[ddlTglEndWork]','[ddlBlnEndWork]',
                '[ddlThnEndWork]');
    start_work_edu = ('[ddlTglStartEdu]',
                '[ddlBlnStartEdu]',
                '[ddlThnStartEdu]');
    end_word_edu = ('[ddlTglEndEdu]','[ddlBlnEndEdu]',
                '[ddlThnEndEdu]');
    address = ('[txtJalan]',' [ddlKota]',' [txtKodepos]',
                ' [ddlPropinsi]');
    branch_no = getBranchNo('[ddlCabang]') ;
    cust_care_no = getCustNo();
```

```
bank_account = ('[txtNoRek]','[txtNamaBank]',
               '[txtNamaCabang]');
work_exp = ('[txtNamaPerusahaan]','[txtJabatan]',
               [start_work_date],[end_work_date]);
education = ('[txtNamaPend]','[ddlBlnStartWork]',
               '[ddlThnStartWork]');

SQL Statement :
UPDATE Agen_TAB
SET Emp_Name = '[txtEmpName]',Branch_No = [branch_no]
,Address = [address],Emp_Password = '[empPassKey]',
Join_Date = [join_date],Status = '[ddlStatus]',
Emp_Phone = '[txtPhoneNo]',
Emp_Birth_Place = '[txtBirthPlace]',
Emp_Birth_Date = [birth_date],Emp_Sex = '[ddlSex]',
Emp_Email = '[txtEmail]',
Emp_Marriage_Status = '[ddlStatusNikah]',
Member_State = '[txtWargaNegara]',
Identity_Card_No = '[txtEmpCardNo]',
Cust_Care_No = [cust_care_no],
Bank_Account = [bank_account],Work_Exp = [work_exp],
Education = [education]
WHERE Emp_No = '[txtEmpNo]';

UPDATE User_TAB
SET Password = '[txtPassKey]'
WHERE User_Name = '[txtEmpNo]';

UPDATE TABLE (SELECT Down_Agen FROM Cust_Care_TAB)
SET Emp_Name = '[txtEmpName]',Join_Date = [join_date];

UPDATE TABLE (SELECT Emp_List FROM Branch_TAB)
SET Emp_Name = '[txtEmpName]',Join_Date = [join_date]
WHERE Emp_No = '[txtEmpNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol edit pada *datagrid* agen yang bersangkutan, maka data tersebut akan tertampil pada *textbox* dan *combo box* sesuai dengan *field*-nya.

4. *User* melakukan perubahan terhadap data agen.
5. *User* menekan tombol *Save*.
6. *Perangkat* lunak akan melakukan pengecekan terhadap data agen yang *user* masukkan, apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Agen_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

Urutan aksi yang terjadi :

1. User melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

Urutan aksi yang terjadi :

1. *User* melakukan klik *link* halaman utama.
2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

Urutan aksi yang terjadi :

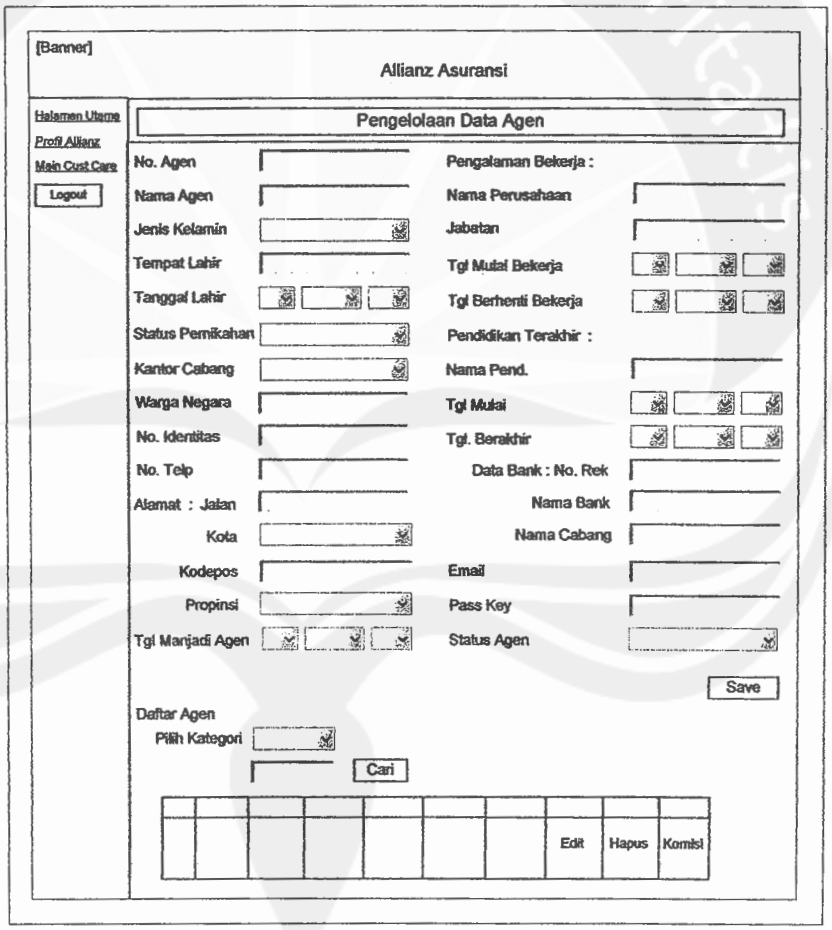
1. *User* melakukan klik *link* profil Allianz.
2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Customer Care*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main customer care*.
  2. Perangkat lunak akan membuka utama *customer care*.

4.16. *Use Case Manage* Agen - *Delete* Agen



Gambar 4.16 Rancangan Halaman *Delete* Agen

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penghapusan data agen. Penghapusan data tersebut akan mempengaruhi 2 tabel yaitu Agen_TAB dan User_TAB.
- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - komisi → tombol untuk menuju ke halaman laporan mengenai komisi yang agen terima.

Event

- *Delete*

```
onClick_Delete()
{
     SQL Statement :
     DELETE FROM Agen_TAB
     WHERE Branch_No = '[txtBranchNo]';

     DELETE FROM User_TAB
     WHERE User_Name = '[txtEmpNo]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol hapus pada baris datagrid sesuai dengan data agen yang akan dihapus.
4. Perangkat lunak akan melakukan penghapusan data agen dari *database* sesuai pilihan *user*.

- Cari

```
onClick_Cari()
{    SQL Statement :
     SELECT * FROM Agen_TAB
     WHERE LOWER ('[ddlKategori]')
     LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

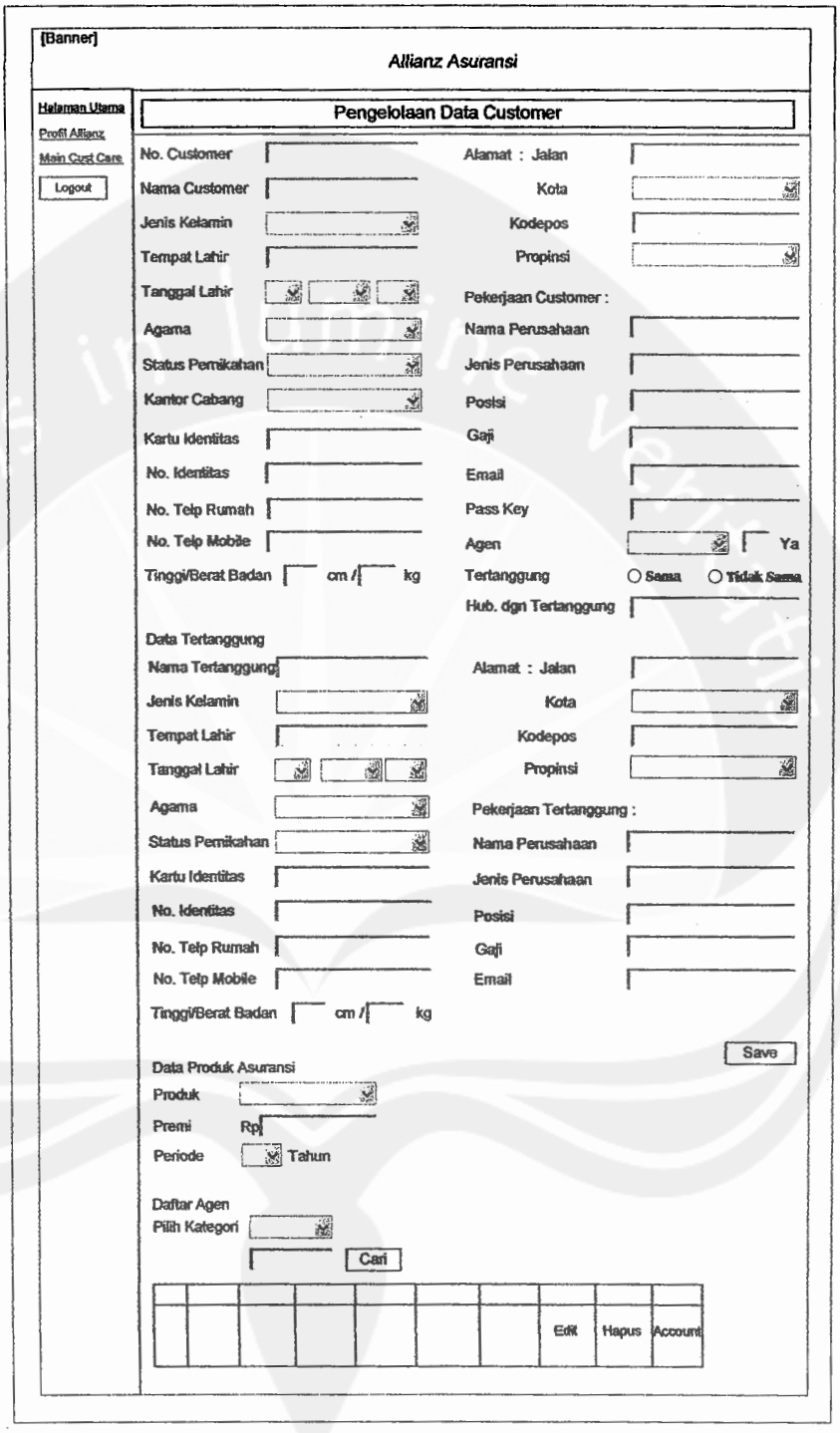
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Admin*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main admin.*
  2. Perangkat lunak akan membuka utama administrator.

4.17. *Use Case Manage Customer – Add Customer*



Gambar 4.17 Rancangan Halaman *Add Customer*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *customer*. Penambahan data tersebut akan mempengaruhi 3 tabel yaitu Customer_TAB, *nested table* Cust_List dari Branch_TAB dan *nested table* Cust_List dari Agen_TAB.
- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - *account* → tombol untuk menuju ke halaman laporan mengenai data pembayaran premi yang telah *customer* lakukan.

Event

- *Save*

```
onClick_Save()
{
    branch_no = getBranchNo('[ddlKantorCabang]');
    tgl_lahir = ('[ddlTglLahir]','[ddlBlnLahir]',
                 '[ddlThnLahir]');
```

```
address = ('[txtJalan]',' [ddlKota]',
            ' [txtKodepos]', ' [ddlPropinsi]');
addressTtg = ('[txtJalanTtg]',' [ddlKotaTtg]',
            ' [txtKodeposTtg]', ' [ddlPropinsiTtg]');
cust_personal_data = ('[txtCustName]','[ddlSex]',
      '[txtTempLahir]',[tgl_lahir],'[txtIdentitas]',
      '[txtNoIdentitas]','[ddlStatusNikah]',
      '[ddlAgama]','[txtTinggi]','[txtBerat]',
      '[txtTlpRmh]','[txtHandphone]','[txtEmail]');
cust_ocupation = ('[txtNamaPerusahaan]',
      '[txtJenisPerusahaan]','[txtPosisi]',
      '[txtGaji]');
cust_personal_dataTtg = ('[txtCustNameTtg]',
      '[ddlSexTtg]','[txtTempLahirTtg]',
       [tgl_lahirTtg],'[txtIdentitasTtg]',
      '[txtNoIdentitasTtg]','[ddlStatusNikahTtg]',
      '[ddlAgamaTtg]','[txtTinggiTtg]',
      '[txtBeratTtg]','[txtTlpRmhTtg]',
      '[txtHandphoneTtg]','[txtEmailTtg]');
Cust_ocupationTtg = ('[txtNamaPerusahaanTtg]',
      '[txtJenisPerusahaanTtg]','[txtPosisiTtg]',
      '[txtGajiTtg]');
product = ('[ddlProduk]','[txtPremi]','[ddlPeriode]');

SQL Statement :
INSERT INTO Customer _TAB (Cust_No,Branch_No,
Cust_Personal_Data,Cust_Ocupation,Address,
Endured_Data,Endured_Relationship,Pass_Key,Product)
VALUES ('[txtCustNo]',[branch_no],
[cust_personal_data], [cust_ocupation],[address],
([cust_personal_dataTtg],[cust_ocupationTtg],
[addressTtg]),'[txtHubTertanggung]','[txtPassKey]',
[product]);

INSERT INTO TABLE (SELECT Cust_List FROM Branch_TAB
WHERE Branch_No = [branch_no])
VALUES ([cust_personal_data],[cust_ocupation],
[address]);

INSERT INTO TABLE (SELECT Cust_List FROM Agen_TAB
WHERE Branch_No = [branch_no])
VALUES ([cust_personal_data],[cust_ocupation],
[address]);

}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *customer* pada masing-masing *textbox* dan *combo box* yang tersedia.

2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *customer* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

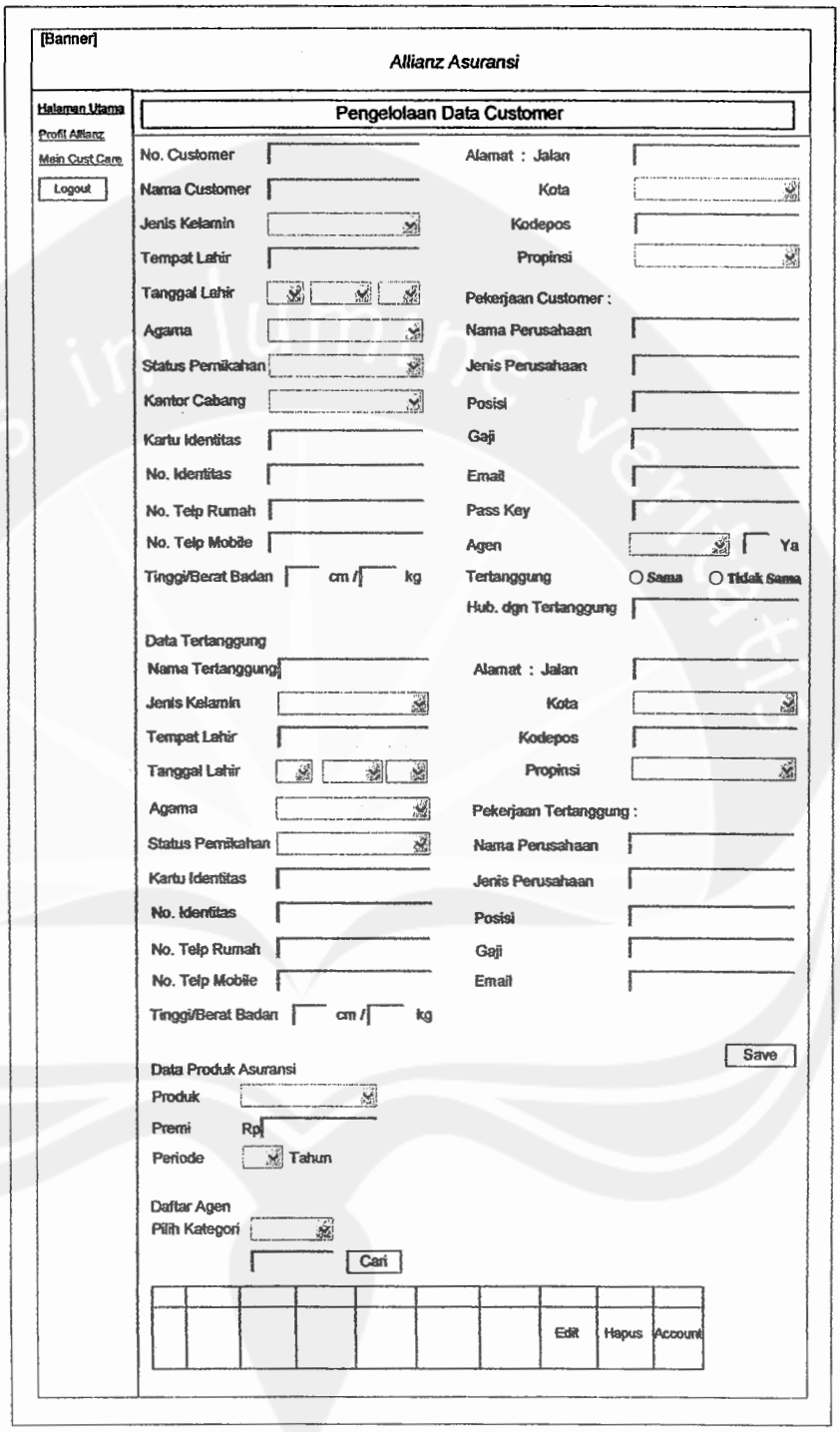
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Customer Care*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main customer care*.
  2. Perangkat lunak akan membuka utama *customer care*.

4.18. *Use Case Manage Customer – Edit* Customer



Gambar 4.18 Rancangan Halaman *Edit Customer*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *customer*. Pengeditan data tersebut akan mempengaruhi 2 tabel, yaitu Cust_Care_TAB dan User_TAB.
- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - *account* → tombol untuk menuju ke halaman laporan mengenai data pembayaran premi yang telah *customer* lakukan.

Event

- *Save*

```
onClick_Save()
{
    branch_no = getBranchNo('[ddlKantorCabang]');
    tgl_lahir = ('[ddlTglLahir]','[ddlBlnLahir]',
                 '[ddlThnLahir]');
    address = ('[txtJalan]',' [ddlKota]',
                ' [txtKodepos]', ' [ddlPropinsi]');
```

```
addressTtg = ('[txtJalanTtg]',' [ddlKotaTtg]',
            ' [txtKodeposTtg]', ' [ddlPropinsiTtg]');
cust_personal_data = ('[txtCustName]','[ddlSex]',
      '[txtTempLahir]',[tgl_lahir],'[txtIdentitas]',
      '[txtNoIdentitas]','[ddlStatusNikah]',
      '[ddlAgama]','[txtTinggi]','[txtBerat]',
      '[txtTlpRmh]','[txtHandphone]','[txtEmail]');
cust_ocupation = ('[txtNamaPerusahaan]',
      '[txtJenisPerusahaan]','[txtPosisi]',
      '[txtGaji]');
cust_personal_dataTtg = ('[txtCustNameTtg]',
      '[ddlSexTtg]','[txtTempLahirTtg]',
       [tgl_lahirTtg],'[txtIdentitasTtg]',
      '[txtNoIdentitasTtg]','[ddlStatusNikahTtg]',
      '[ddlAgamaTtg]','[txtTinggiTtg]',
      '[txtBeratTtg]','[txtTlpRmhTtg]',
      '[txtHandphoneTtg]','[txtEmailTtg]');
Cust_ocupationTtg = ('[txtNamaPerusahaanTtg]',
      '[txtJenisPerusahaanTtg]','[txtPosisiTtg]',
      '[txtGajiTtg]');
produk = ('[ddlProduk]','[txtPremi]','[ddlPeriode]');

SQL Statement :
UPDATE Customer_TAB
SET Branch_No = [branch_no],
Cust_Personal_Data = [cust_personal_data],
Cust_Ocupation = [cust_ocupation],
Address = [address],
Endured_Data = ([cust_personal_dataTtg],
[cust_ocupationTtg],[addressTtg]),
Endured_Relationship = '[txtHubTertanggung]',
Pass_Key = '[txtPassKey]';

UPDATE TABLE (SELECT Cust_List FROM Branch_TAB
WHERE Branch_No = [branch_no])
SET Cust_Personal_Data = cust_personal_data,
Cust_Ocupation = cust_ocupation,
Address = address;

UPDATE TABLE (SELECT Cust_List FROM Agen_TAB
WHERE Branch_No = [branch_no])
SET Cust_Personal_Data = cust_personal_data,
Cust_Ocupation = cust_ocupation,
Address = address;
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.

2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol edit pada *datagrid customer* yang bersangkutan, maka data tersebut akan tertampil pada *textbox* dan *combo box* sesuai dengan *field*-nya.
4. *User* melakukan perubahan terhadap data *customer*.
5. *User* menekan tombol *Save*.
6. *Perangkat* lunak akan melakukan pengecekan terhadap data *customer* yang *user* masukkan, apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- Cari

```
onClick_Cari()
{
    SQL Statement :
    SELECT * FROM Customer_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

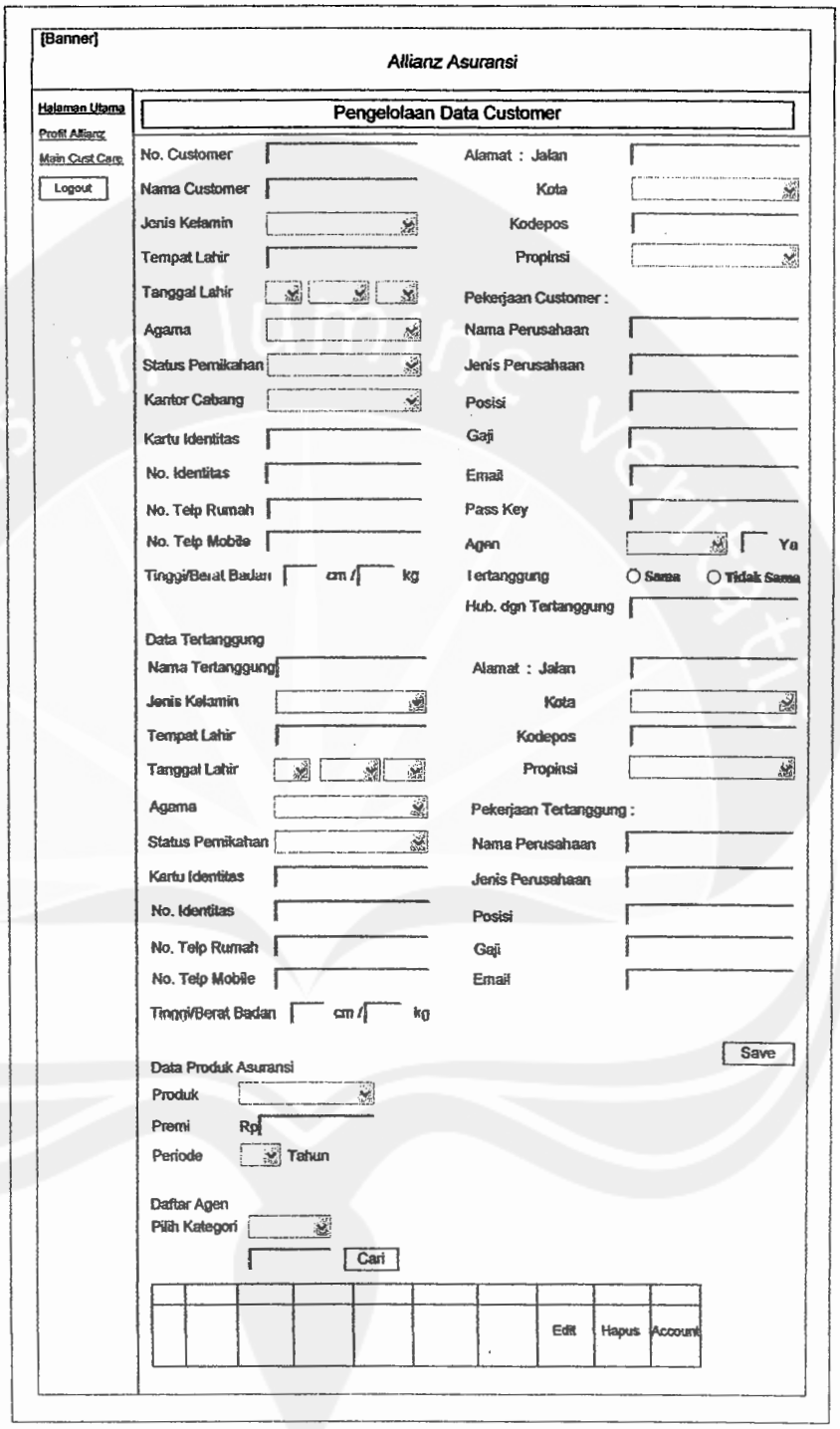
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Customer Care*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main customer care*.
  2. Perangkat lunak akan membuka utama *customer care*.

4.19. *Use Case Manage Customer - Delete Customer*



Gambar 4.19 Rancangan Halaman *Delete Customer*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penghapusan data *customer*. Penghapusan data tersebut akan mempengaruhi 5 tabel yaitu Customer_TAB, *nested table* Cust_List dari Branch_TAB, *nested table* Cust_List dari Agen_TAB, *table* Cust_List dari Cust_Care_TAB, dan User_TAB.
- Pada halaman ini, terdapat 6 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *customer care*.
  - cari → tombol untuk melakukan pencarian data *customer care*.
  - edit → tombol untuk memulai proses *editing* terhadap data *customer care*.
  - hapus → tombol untuk melakukan penghapusan terhadap data *customer care* sesusi dengan pilihan *user*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - *account* → tombol untuk menuju ke halaman laporan mengenai data pembayaran premi yang telah *customer* lakukan.

Event

- *Delete*

```
onClick_Delete()
{
    branch_no = getBranchNo(current_user.branch);
    agenNo = getAgenEmpNo('[ddlAgen]');
    user_name = getUserName('[txtCustNo]');
```

```
SQL Statement :
DELETE FROM TABLE (SELECT Cust_List FROM Branch_TAB
WHERE Branch_No = [branch_no])
WHERE Cust_No = '[txtCustNo]';

DELETE FROM TABLE (SELECT Cust_List FROM Cust_Care_TAB
WHERE Emp_No = [current_user.emp_no])
WHERE Cust_No = '[txtCustNo]';

DELETE FROM TABLE (SELECT Cust_List FROM Agen_TAB
WHERE Emp_No = [agenNo])
WHERE Cust_No = '[txtCustNo]';

DELETE FROM Cust_Care_TAB
WHERE Cust_No = '[txtCustNo]';

DELETE FROM User_TAB
WHERE User_Name = [user_name];
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.
3. *User* klik tombol hapus pada baris datagrid sesuai dengan data agen yang akan dihapus.
4. Perangkat lunak akan melakukan penghapusan data *customer* dari *database* sesuai pilihan *user*.

- Cari

```
onClick_Cari()
{   SQL Statement :
    SELECT * FROM Customer_TAB
    WHERE LOWER ('[ddlKategori]')
    LIKE LOWER ('[ddlKeyword]');
}
```

Urutan aksi yang terjadi :

1. *User* memilih kategori pencarian dan memasukkan *keyword* di *textbox* yang telah disediakan.
2. *User* klik tombol cari, maka data yang sesuai dengan kategori dan *keyword* pencarian akan tertampil di *datagrid*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

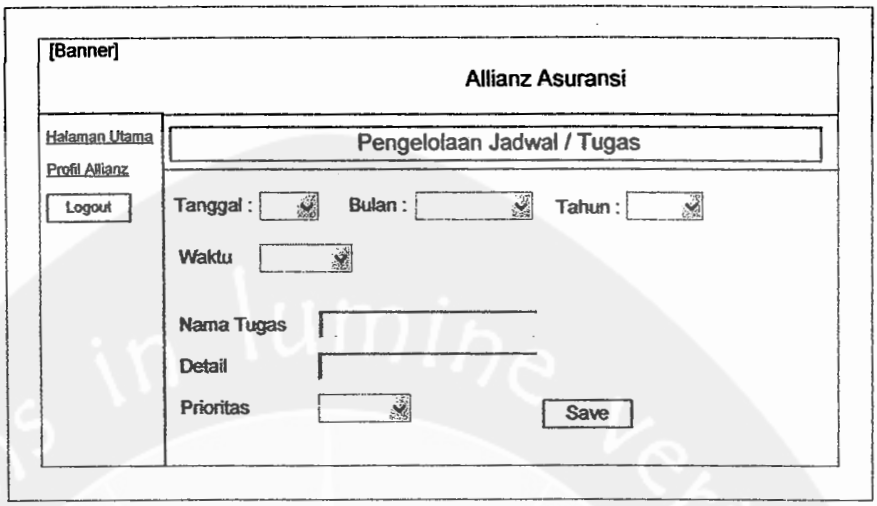
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

- *Link Main Customer Care*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link main customer care*.
  2. Perangkat lunak akan membuka utama *customer care*.

4.20. *Use Case Manage Task – Add Task*



Gambar 4.20 Rancangan Halaman *Add Task*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penambahan data *task*. Penambahan data tersebut akan mempengaruhi sebuah *nested table* yaitu Task.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *task*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    task_date = ('[txtTgl]','[txtBln]','[txtThn]');
    If user_type = 'Agen'
        {
            INSERT INTO TABLE (SELECT Task
```

```
                    FROM Agen_TAB
                    WHERE Emp_No = current_user.emp_no)
                    VALUES (Task_TY('[txtNamaTgs]',
                    '[txtDetail]',[task_date],'[ddlTime]',
                    '[ddlPrioritas]'));
            }
      Else if user_type = 'Customer Care'
            {
                    INSERT INTO TABLE (SELECT Task
                    FROM Cust_Care_TAB
                    WHERE Emp_No = current_user.emp_no)
                    VALUES (Task_TY('[txtNamaTgs]',
                    '[txtDetail]',[task_date],'[ddlTime]',
                    '[ddlPrioritas]'))
            }
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *task* pada masing-masing *textbox* dan *combo box* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *task* yang *user* masukkan, apabila *valid*, maka data tersebut akan ditambahkan ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

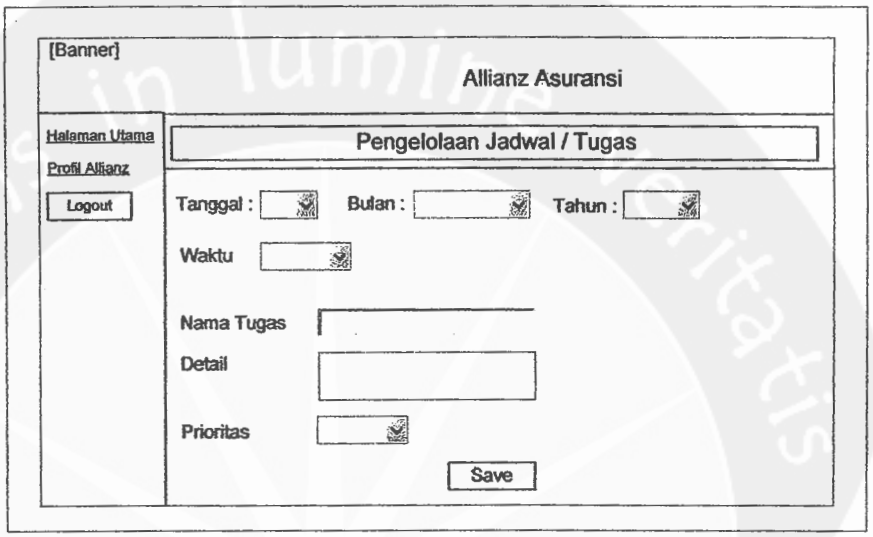
- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz
  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.21. *Use Case Manage Task – Edit Task*



Gambar 4.21 Rancangan Halaman *Edit Task*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan pengeditan data *task*. Pengeditan data tersebut akan mempengaruhi sebuah *nested table* yaitu Task.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *task*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
     task_date = ('[txtTgl]','[txtBln]','[txtThn]');
     If user_type = 'Agen'
          {
               UPDATE TABLE (SELECT Task
               FROM Agen_TAB
               WHERE Emp_No = current_user.emp_no)
               SET Task = (('[txtNamaTgs]',
               '[txtDetail]',[task_date],'[ddlTime]',
               '[ddlPrioritas]'));
          }
     Else if user_type = 'Customer Care'
          {
               UPDATE TABLE (SELECT Task
               FROM Cust_Care_TAB
               WHERE Emp_No = current_user.emp_no)
               SET Task = (('[txtNamaTgs]',
               '[txtDetail]',[task_date],'[ddlTime]',
               '[ddlPrioritas]'));
          }
}
```

Urutan aksi yang terjadi :

1. *User* mengupdate semua data *task* pada masing-masing *textbox* dan *combo box* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan pengecekan terhadap data *task* yang user masukkan, apabila *valid*, maka data tersebut akan di-*update* ke dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

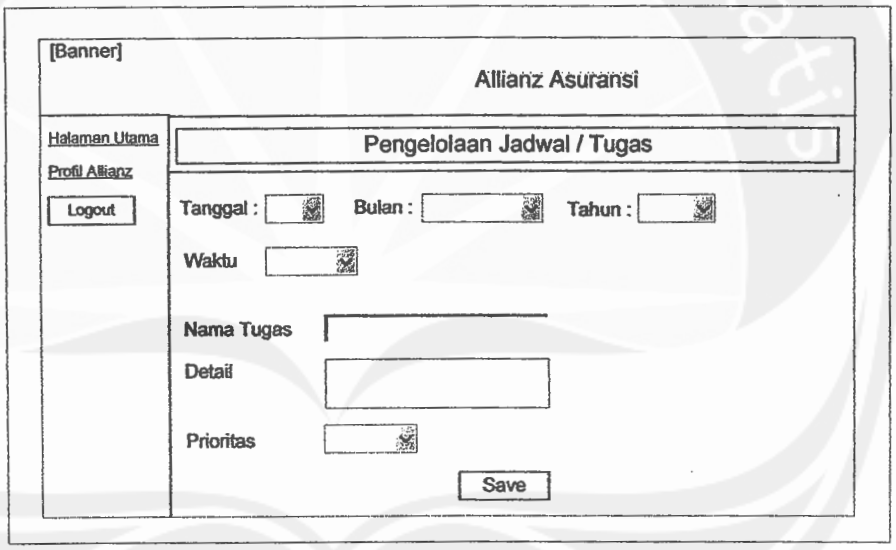
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.22. *Use Case Manage Task – Delete Task*



Gambar 4.22 Rancangan Halaman *Delete Task*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan penghapusan data *task*. Penghapusan data tersebut akan mempengaruhi sebuah *nested table* yaitu Task.

- Pada halaman ini, terdapat 2 buah tombol, yaitu:
  - *save* → tombol untuk mengeksekusi penambahan ataupun perubahan yang dilakukan terhadap data *task*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
     task_date = ('[txtTgl]','[txtBln]','[txtThn]');
     If user_type = 'Agen'
          {
               DELETE FROM TABLE (SELECT Task
               FROM Agen_TAB
               WHERE Emp_No = current_user.emp_no)
               WHERE Task_Name = '[txtNamaTgs]'
               AND Task_Date = [task_date];
          }
     Else if user_type = 'Customer Care'
          {
               DELETE FROM TABLE (SELECT Task
               FROM Cust_Care_TAB
               WHERE Emp_No = current_user.emp_no)
               WHERE Task_Name = '[txtNamaTgs]'
               AND Task_Date = [task_date];
          }
}
```

Urutan aksi yang terjadi :

1. *User* menghapus semua data *task* pada masing-masing *textbox* dan *combo box* yang tersedia.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan menghapus data *task* yang bersangkutan dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

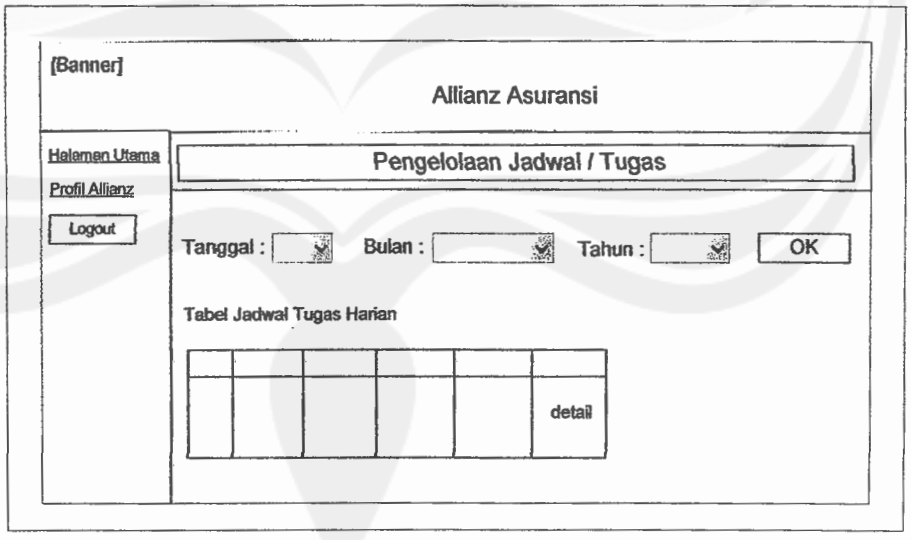
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.23. *Use Case Manage Task – Display Task*



Gambar 4.23 Rancangan Halaman *Display Task*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melihat atau mendapatkan informasi mengenai daftar *task*.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - OK → tombol untuk mengeksekusi pencarian terhadap hari *task* tersebut harus dilaksanakan.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - *detail* → untuk menuju ke halaman penambahan, pengeditan, dan menghapusan data *task*, sesuai dengan waktu yang dipilih.

Event

- OK

```
onClick_OK()
{
    SQL Statement :
    SELECT Task_Name, Task_Detail, Task_Priority, Time
    FROM Agen_TAB A1, TABLE(A1.Task) A2
    WHERE TO_CHAR(Task_Date,'DD') = '[ddlTgl]'
    AND TO_CHAR(Task_Date,'Mon') =  '[ddlBln]'
    AND TO_CHAR(Task_Date,'YYYY') = '[ddlThn]';
}
```

Urutan aksi yang terjadi :

1. *User* memilih tanggal, bulan, dan tahun pada masing-masing *drop down list* yang disediakan.
2. *User* menekan tombol OK.
3. Perangkat lunak akan menampilkan data *task* sesuai masukan *user* di atas dalam *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

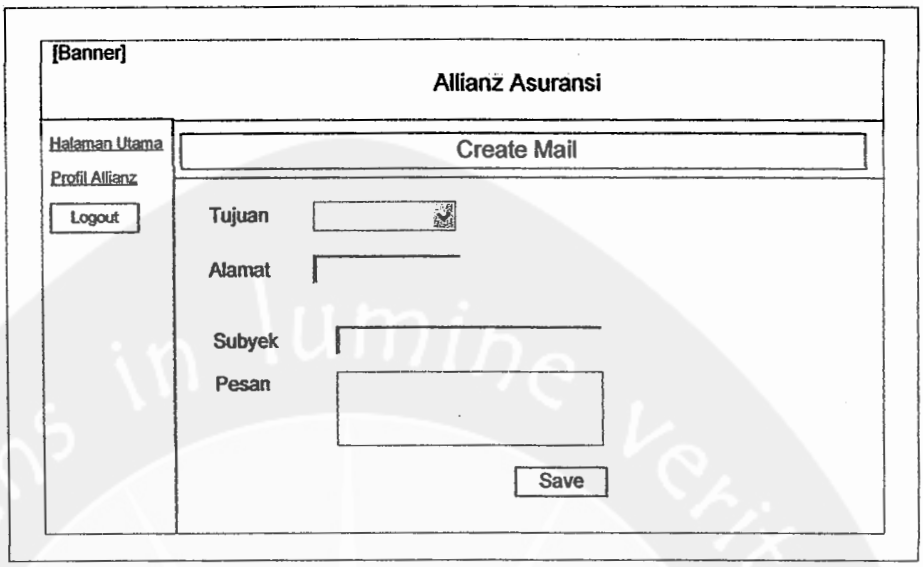
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.24. *Use Case Manage Mail – Create Mail*



Gambar 4.24 Rancangan Halaman *Create Mail*

Deskripsi

- Halaman ini digunakan oleh *user* untuk membuat dan mengirimkan *mail*.
- Pada halaman ini, terdapat 2 buah tombol, yaitu:
  - *Save* → tombol untuk mengeksekusi pembuatan dan pengiriman *mail*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    date = ('[ddlTgl]','[ddlBln]','[ddlThn]');
    current_user_tab = getCurrentUserTAB([current_user]);
    if tujuan = 'Agen'
        {
            SQL Statement :
```

```
                    INSERT INTO TABLE (SELECT MAIL FROM Agen
                    WHERE Emp_No = '[txtTujuan]')
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Inbox',[current_user]);

                    INSERT INTO TABLE (SELECT MAIL FROM
                    [current_user_tab]
                    WHERE Emp_No = [current_user])
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Outbox','[txtTujuan]');
              }
        else if tujuan = 'Cust_Care'
              {
                    SQL Statement :
                    INSERT INTO TABLE (SELECT MAIL FROM
                    Cust_Care_TAB
                    WHERE Emp_No = '[txtTujuan]')
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Inbox',[current_user]);

                    INSERT INTO TABLE (SELECT MAIL FROM
                    [current_user_tab]
                    WHERE Emp_No = [current_user])
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Outbox','[txtTujuan]');
              }

        else if tujuan = 'Central_Mgr'
              {     SQL Statement :
                    INSERT INTO TABLE (SELECT MAIL FROM
                    Central_Mgr_TAB
                    WHERE Emp_No = '[txtTujuan]')
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Inbox',[current_user]);

                    INSERT INTO TABLE (SELECT MAIL FROM
                    [current_user_tab]
                    WHERE Emp_No = [current_user])
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Outbox','[txtTujuan]');
              }
        else if tujuan = 'Customer'
              {     SQL Statement :
                    INSERT INTO TABLE (SELECT MAIL FROM
                    Customer_TAB
                    WHERE Emp_No = '[txtTujuan]')
                    VALUES (date,'[txtSubyek]','[txtPesan]',
                    'Inbox',[current_user]);

                    INSERT INTO TABLE (SELECT MAIL FROM
                    [current_user_tab]
                    WHERE Emp_No = [current_user])
                    VALUES (date,'[txtSubyek]','[txtPesan]',
```

```
                'Outbox','[txtTujuan]');
        }
    else if tujuan = 'Admin'
        {   SQL Statement :
            INSERT INTO TABLE (SELECT MAIL FROM
            Admin_TAB
            WHERE Emp_No = '[txtTujuan]')
            VALUES (date,'[txtSubyek]','[txtPesan]',
            'Inbox',[current_user]);

            INSERT INTO TABLE (SELECT MAIL FROM
            [current_user_tab]
            WHERE Emp_No = [current_user])
            VALUES (date,'[txtSubyek]','[txtPesan]',
            'Outbox','[txtTujuan]');
        }
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan semua data *mail* yang diperlukan.
2. *User* menekan tombol OK.
3. Perangkat lunak akan mengirimkan mail sesuai dengan tujuan yang *user* masukkan.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

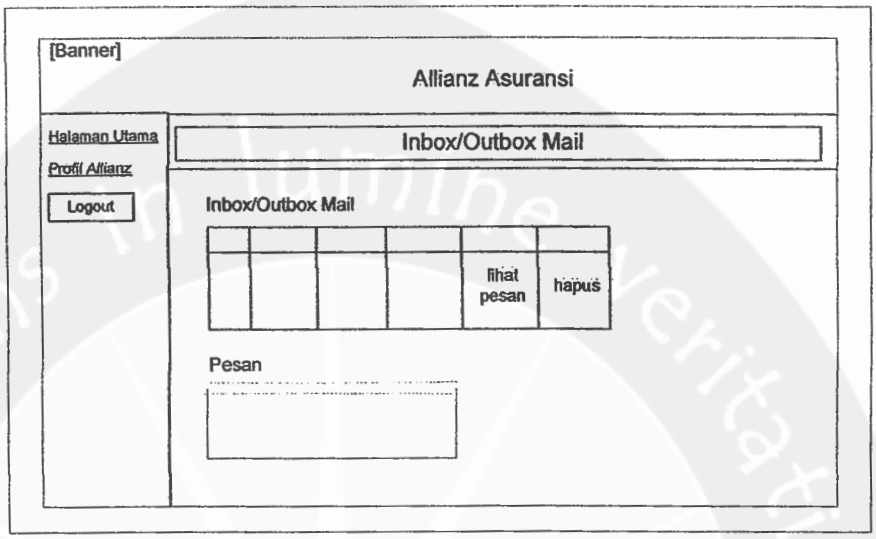
- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.

2. Perangkat lunak akan membuka halaman profil Allianz.

4.25. *Use Case Manage Mail - Delete Mail*



Gambar 4.25 Rancangan Halaman *Delete Mail*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melihat detail/isi dan menghapus *mail*.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - lihat pesan → tombol untuk melihat detail atau isi *mail*.
  - hapus → tombol untuk melakukan penghapusan terhadap *mail* yang dipilih pada *datagrid*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- Hapus

```
onClick_Hapus()
{    subyek = getSubject();
     isi = getIsi();
```

```
tipe = getType();
If current_user = 'Admin'
      {
            SQL Statement :
            DELETE    FROM    TABLE    (SELECT    MAIL
            FROM Mail_Admin_TAB)
            WHERE Subject = subyek AND content = isi
            AND type = tipe;
      }
Else if current_user = 'Agen'
      {
            SQL Statement :
            DELETE    FROM    TABLE    (SELECT    MAIL
            FROM Agen_TAB)
            WHERE Subject = subyek AND content = isi
            AND type = tipe;
      }
Else if current_user = 'Cust_Care'
      {
            SQL Statement :
            DELETE    FROM    TABLE    (SELECT    MAIL
            FROM Cust_Care_TAB)
            WHERE Subject = subyek AND content = isi
            AND type = tipe;
      }

Else if current_user = 'Central_Manager'
      {
            SQL Statement :
            DELETE    FROM    TABLE    (SELECT    MAIL
            FROM Central_Mgr_TAB)
            WHERE Subject = subyek AND content = isi
            AND type = tipe;
      }
Else if current_user = 'Customer'
      {
            SQL Statement :
            DELETE    FROM    TABLE    (SELECT    MAIL
            FROM Customer_TAB)
            WHERE Subject = subyek AND content = isi
            AND type = tipe;
      }
}
```

Urutan aksi yang terjadi :

1. *User* menentukan data *mail* yang ingin dihapus.
2. *User* menekan tombol hapus.
3. Perangkat lunak akan melakukan penghapusan *mail* dari *database*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

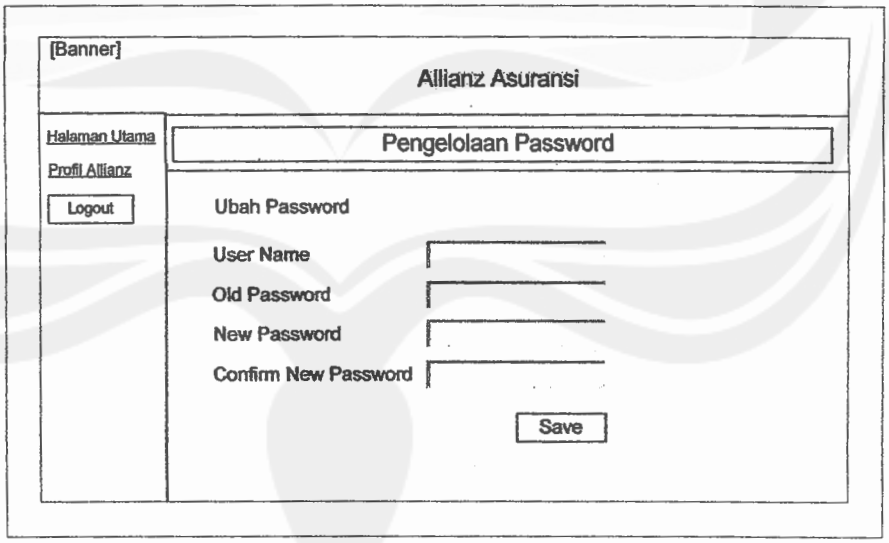
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.26. *Use Case Manage Password*



Gambar 4.26 Rancangan Halaman *Manage Password*

Deskripsi

- Halaman ini digunakan oleh *user* untuk melakukan perubahan *password*.
- Pada halaman ini, terdapat 3 buah tombol, yaitu:
  - lihat pesan → tombol untuk melihat detail atau isi *mail*.
  - hapus → tombol untuk melakukan penghapusan terhadap *mail* yang dipilih pada *datagrid*.
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.

Event

- *Save*

```
onClick_Save()
{
    SQL Statement :
    UPDATE User_TAB
    SET Password = '[txtNewPassword]';
}
```

Urutan aksi yang terjadi :

1. *User* memasukkan password, password baru dan confirm password baru.
2. *User* menekan tombol *Save*.
3. Perangkat lunak akan melakukan meng-*update* *database*.

- *Logout*

Urutan aksi yang terjadi :

1. *User* melakukan klik tombol *Logout*.
2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

Urutan aksi yang terjadi :

Event

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

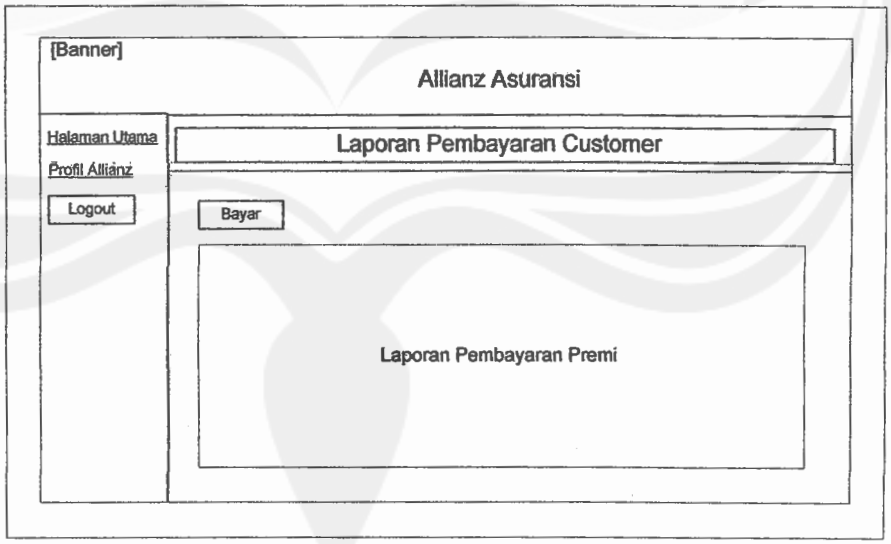
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.28. *Use Case Customer Report*



Gambar 4.28 Rancangan Halaman *Customer Report*

Deskripsi

- Halaman ini digunakan oleh *user* untuk mendapatkan laporan mengenai daftar pembayaran premi oleh *customer*.
- Pada halaman ini, terdapat sebuah tombol, yaitu:
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - Bayar → tombol untuk menuju halaman pembayaran premi *customer*.

Event

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.
- *Link* Halaman Utama

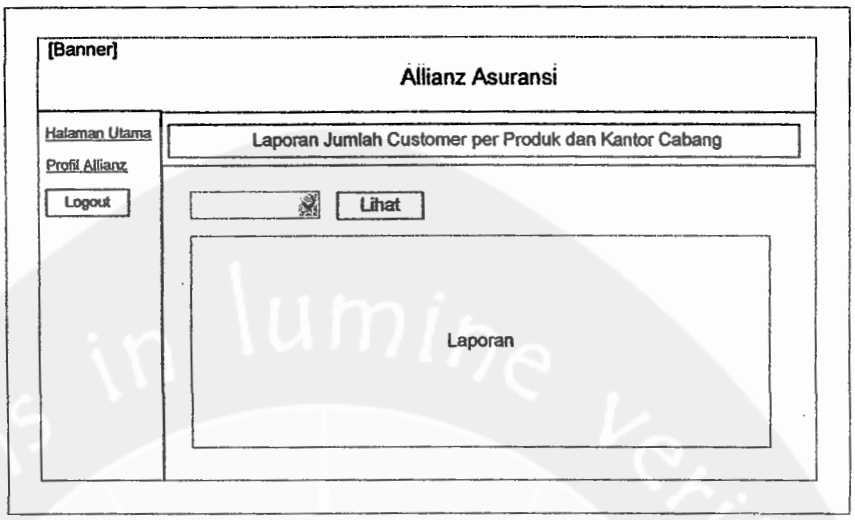
  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.
- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.29. *Use Case Product Report*



Gambar 4.29 Rancangan Halaman *Product Report*

Deskripsi

- Halaman ini digunakan oleh *user* untuk mendapatkan laporan mengenai perbandingan (dalam bentuk grafik) jumlah customer dilihat dari produk yang diambil dan kantor cabang.
- Pada halaman ini, terdapat sebuah tombol, yaitu:
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - Lihat → tombol menampilkan laporan dengan memilih kantor cabang terlebih dahulu.

Event

- Lihat

```
onClick_Lihat()
{
    ReportProduk.SelectionFormula =
    "{Product_Report.Branch_Name} = '[ddlKantorCabang]'
}
```

Urutan aksi yang terjadi :

1. *User* memilih kantor cabang pada *drop down list* yang ada.
2. *User* klik tombol Lihat.
3. Perangkat lunak akan menampilkan laporan sesuai dengan pemilihan kantor cabang oleh *user*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

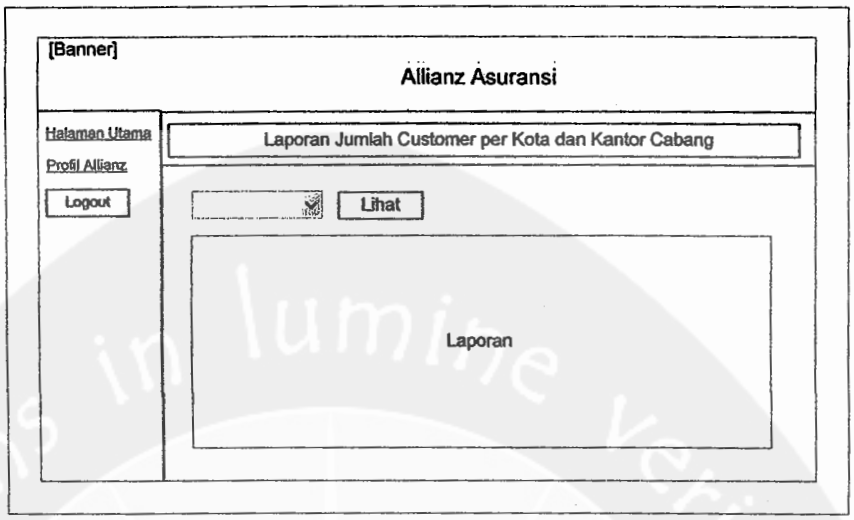
  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.

4.30. *Use Case Branch Report*



Gambar 4.30 Rancangan Halaman *Branch Report*

Deskripsi

- Halaman ini digunakan oleh *user* untuk mendapatkan laporan mengenai perbandingan (dalam bentuk grafik) jumlah customer dilihat dari kota dan kantor cabang.
- Pada halaman ini, terdapat sebuah tombol, yaitu:
  - *Logout* → tombol untuk melakukan *Logout* dan kembali ke halaman utama.
  - Lihat → tombol menampilkan laporan dengan memilih kota terlebih dahulu.

Event

- Lihat

```
onClick_Lihat()
{
    ReportBranch.SelectionFormula =
    "{Branch_Report.City} = '[ddlKota]'
}
```

Urutan aksi yang terjadi :

1. *User* memilih kota pada *drop down list* yang ada.
2. *User* klik tombol Lihat.
3. Perangkat lunak akan menampilkan laporan sesuai dengan pemilihan kota oleh *user*.

- *Logout*

  Urutan aksi yang terjadi :

  1. *User* melakukan klik tombol *Logout*.
  2. Status *user* menjadi *Logout* dan perangkat lunak akan membuka halaman utama Allianz.

- *Link* Halaman Utama

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* halaman utama.
  2. Perangkat lunak akan membuka halaman utama Allianz.

- *Link* Profil Allianz

  Urutan aksi yang terjadi :

  1. *User* melakukan klik *link* profil Allianz.
  2. Perangkat lunak akan membuka halaman profil Allianz.