BAB V

KESIMPULAN DAN SARAN

V.1. Kesimpulan

Setelah mengamati dan membuat Aplikasi Retail Komputer dengan Teknologi AJAX (ARKAJAX), maka dapat ditarik kesimpulan, yaitu :

1. Aplikasi retail komputer dengan teknologi AJAX telah dibangun dengan baik
2. Saat halaman pertama kali dipanggil dan ditampilkan ke *web browser*, halaman *web* yang menggunakan teknologi AJAX cenderung membutuhkan waktu yang lebih banyak dibandingkan halaman *web* yang tidak menggunakan AJAX
3. Dengan menggunakan teknologi AJAX, waktu yang diperlukan untuk memproses sebuah *request* lebih sedikit dibandingkan pemrosesan *request* tanpa menggunakan teknologi AJAX
4. Semakin banyak *request* yang terjadi dalam satu *event*, maka total waktu yang dibutuhkan untuk menyelesaikan satu *event* semakin besar
5. ARKAJAX dibuat dengan menggunakan framework Ajax.NET yang *support* untuk ASP.NET 2003, yang memiliki kelemahan, hanya bisa menjalankan AJAX di *web browser Internet Explorer* 6

V.2. Saran

Saran-saran yang dapat diberikan berdasarkan analisa dan pembuatan laporan ini adalah :

1. Aplikasi Retail Komputer dengan Teknologi AJAX dapat dikembangkan lebih lanjut lagi sehingga dapat dijalankan di semua *web browser*, kecuali *web browser* yang tidak mendukung *javascript*
2. Karena program ini masih jauh dari sempurna, penulis menyarankan sistem ini dapat lebih dikembangkan lagi tampilannnya sehingga menyerupai tampilan aplikasi *dekstop*

Daftar Pustaka

Agung, RJB Wahju, *Diktat Mata Kuliah: Basis Data,* Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Atma Jaya Yogyakarta

Duthie, G Andrew, Microsoft ASP.NET Step By Step, PT Elex Media Komputindo Gramedia, 2003

Rickyanto, Isak, Membuat Aplikasi Web Dengan ASP.NET, PT Elex Media Komputindo Gramedia, 2003

Ang, Olivia, Tip Dan Trik Rahasia Mahir Menguasai ASP.NET,*2004*

Oestereich, Bernd, Developing Sofware With UML Object Oriented Analysis And Design In Practice, Addison-Wesley, 2002

Suhendar, A., Hariman Gunadi, S.Si.,MT, *Visual Modeling Menggunakan UML dan Rational Rose*, Informatika Bandung, 2002

Holzner, Steve, Ph.D, AJAX For Dummies, Wiley Pubhising Inc, 2006

Telerik, Using AJAX in Web Applications A Practical Guide for ASP.NET Developers, Telerik.com, 2005

Spesifikasi Kebutuhan Perangkat Lunak

Aplikasi Retail Komputer Berbasis Web Dengan
Menggunakan Teknologi AJAX
(ARKAJAX)

Disusun oleh :

Kuntoro Haryatmoko

01 07 02979

Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Atma Jaya Yogyakarta

DAFTAR PERUBAHAN

Revisi	Deskripsi
1	

INDEX TGL	-	A	B	C	D	E	F
Ditulis oleh							
Diperiksa oleh							
Disetujui oleh							

NOTASI DOKUMEN

Notasi yang digunakan dalam dokumen ini adalah sebagai berikut :

- Teks normal ditulis dalam *font* Times New Roman 12 pt, *plain*.
- Teks yang ditulis dalam ***font bold*** merupakan teks yang mengacu pada bab, sub-bab, gambar, atau tabel dalam dokumen ini.
- Teks yang ditulis dalam *font* Courier New merupakan teks yang mengacu pada model, diagram, atau file yang disebutkan dalam dokumen ini.

DAFTAR ISI

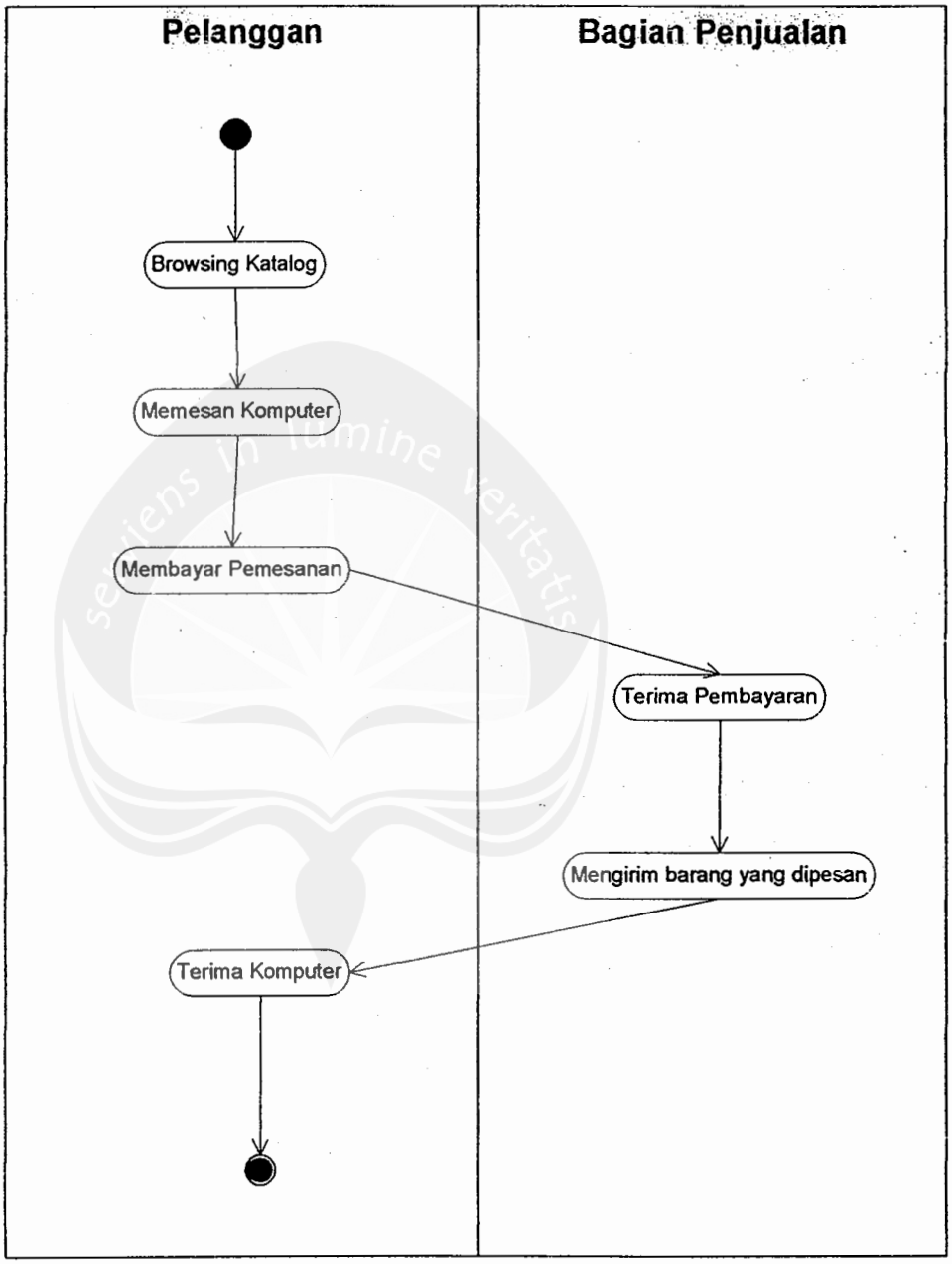
1. Pendahuluan

1.1. Tujuan

Dokumen ini berisi tentang penjelasan Spesifikasi Kebutuhan Perangkat Lunak (SKPL). Dokumen SKPL ini merupakan dokumen spesifikasi kebutuhan perangkat lunak untuk sistem ARKAJAX yang akan dibangun. Dokumen ini digunakan oleh pengembang perangkat lunak sebagai acuan teknis untuk pengembangan perangkat lunak. Dalam SKPL ini akan dijelaskan mengenai kebutuhan-kebutuhan yang harus tersedia agar perangkat lunak yang diharapkan dapat terwujud.

1.2. Lingkup Masalah

Gambar dibawah menunjukkan gambaran sistem penjualan komputer secara *online*:



Gambar 1.1 Activity Diagram untuk ARKAJAX

Kebutuhan manusia akan komputer sebagai alat bantu untuk menyelesaikan pekerjaan maupun sebagai hiburan *digital* belakangan ini

menjadi meningkat pesat. Permintaan akan komputer biasanya datang dari kalangan pebisnis maupun kalangan akademisi.

Tingginya permintaan akan komputer ini merupakan lahan basah bagi para pebisnis retail komputer untuk meraih untung yang besar. Untuk dapat meraih untung yang besar, maka perlu untuk menjaring pelanggan sebanyak-banyaknya. Untuk memperoleh pelanggan yang banyak maka perlu adanya ekspansi pasar. Hal ini dimaksudkan agar retail komputer dapat dikenal orang banyak, dan mampu melayani permintaan pelanggan yang lokasinya berjauhan.

Untuk dapat melayani pelanggan yang lokasinya berjauhan maka dibangunlah aplikasi retail komputer berbasis *web*, dimana *retail* komputer berbasis *web* ini dapat mewakili retail komputer yang sesungguhnya, sehingga *retail* komputer tersebut dapat dikenal diseluruh dunia dan mampu untuk melayani permintaan pelanggan jauh lokasinya.

Secara umum permasalahan tentang aplikasi *retail* komputer berbasis *web* muncul ketika seorang klien merasa kesal karena harus lama menunggu informasi yang diperoleh melalui aplikasi retail komputer berbasis web menampilkan halaman web yang diminta oleh pelanggan. Untuk itu diperlukan suatu cara agar permasalahan yang terjadi disisi klien tersebut dapat diatasi. Dengan menggunakan teknologi AJAX maka permasalahan tentang lamanya pelanggan harus menunggu munculnya suatu halaman web disisi klien yang diminta dapat teratasi.

Gambar 1.1 menggambarkan proses-proses yang terjadi disisi klien ketika seorang klien memesan komputer melalui web. Pemesanan komputer dimulai oleh klien yang melakukan proses `browsing katalog`, proses ini memungkinkan seorang klien untuk mencari mencari informasi tentang pc, server, atau notebook yang diminta. Jika seorang klien hendak memesan komputer maka proses `memesan komputer` akan dijalankan dilanjutkan dengan proses `membayar pemesanan`. Proses ini memungkinkan seorang klien untuk membayar

pemesanan pc, server atau notebook via web, dimana proses pembayaran dilakukan dengan menggunakan kartu kredit. Selanjutnya proses akan mengalir ke bagian penjualan yaitu proses terima pembayaran dilanjutkan dengan proses mengirim barang yang dipesan. Kemudian proses akan mengalir ke klien yaitu proses terima komputer, yang merupakan akhir dari penjualan komputer secara online.

Pada sistem ARKAJAX proses-proses yang akan dibuat dengan menggunakan teknologi AJAX antara lain: proses browsing katalog. Proses browsing katalog digunakan untuk menampilkan product yang dijual oleh retail komputer. Untuk dapat menampilkan product, maka pelanggan memasukkan jenis product serta nama product, kemudian sistem akan menampilkan product ke halaman web. Teknologi AJAX pada proses browsing katalog akan digunakan untuk menampilkan informasi product yang telah ditentukan oleh pelanggan. Dengan menggunakan teknologi AJAX, diharapkan informasi product tersebut dapat ditampilkan dengan cepat.

1.3. Definisi Istilah dan Singkatan

Definisi, istilah dan singkatan yang digunakan dalam dokumen ini mengacu pada **Apendiks A : Daftar Istilah dan Singkatan.**

1.4. Referensi

Referensi yang digunakan dalam pembuatan dokumen ini adalah :

- Dewi, Findra Kartika Sari. Spesifikasi kebutuhan perangkat lunak Fine generator of scheduler. Universitas Atma Jaya Yogyakarta, 2005.
- Oestereich, Bernd. Developing Software With UML Object-Oriented And Design in Practice, Addison-Wesley, 2001

- Singgihraharja, Timotius Pamungkas. Java Program Automatic Gradding Tool. Universitas Atma Jaya Yogyakarta, 2005.
- Agung W, RJB Wahju. Diktat Mata Kuliah Basis Data. Universitas Atma Jaya Yogyakarta, 2001

1.5. Deskripsi Umum (Overview)

Dokumen SKPL ini dibagi menjadi empat bab. Bab pertama adalah **Pendahuluan,** yang berisi tentang deskripsi dokumen. Bab kedua adalah **Deskripsi Keseluruhan,** yang berisi penjelasan secara umum mengenai sistem yang akan dikembangkan meliputi fungsi-fungsi dari sistem, karakteristik pengguna, batasan dan asumsi yang diambil dalam pengembangan perangkat lunak. Bab ketiga adalah **Spesifikasi Rinci Kebutuhan,** yang berisi penjelasan tentang kebutuhan sistem yang akan dikembangkan secara lebih rinci. Bab keempat adalah **Realisasi Use Case**, yang berisi realisasi use case dalam tahap analisis (konseptual), yang akan digunakan sebagai dasar realisasi use case pada tahap desain.

2. Deskripsi Keseluruhan

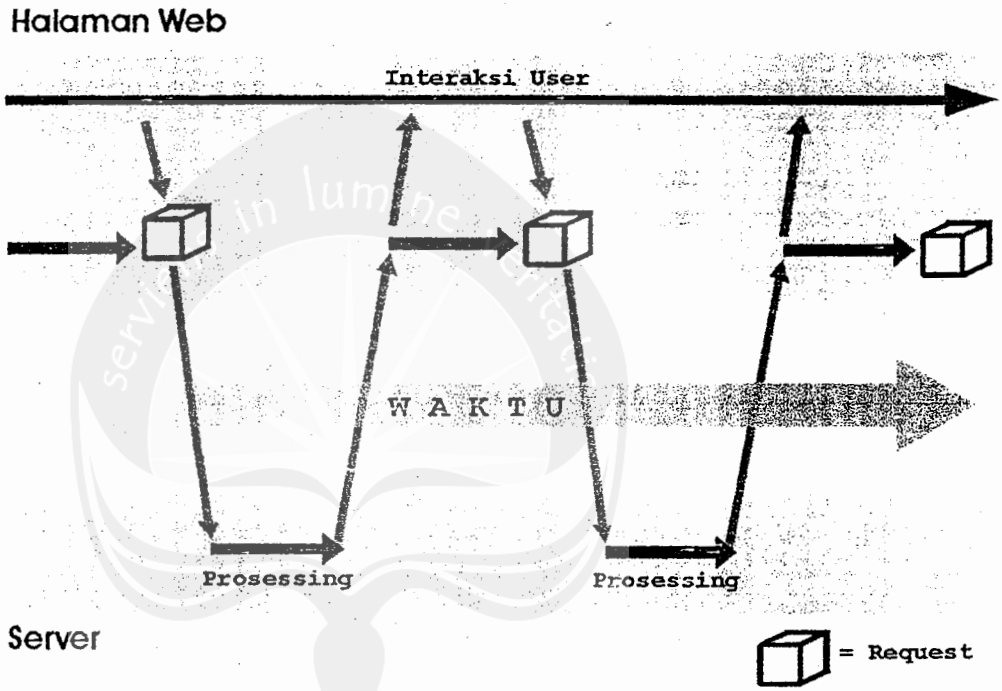
2.1. Perspektif Produk

Aplikasi retail komputer berbasis web dengan menggunakan teknologi AJAX dibangun untuk retail komputer dalam meningkatkan kualitas pelayanan terhadap pelanggannya. Sedangkan bagi klien aplikasi retail komputer berbasis web yang dibangun dengan menggunakan teknologi AJAX, dapat mengurangi beban stres karena lamanya akses terhadap informasi yang diperlukan.

Aplikasi retail komputer berbasis web yang dibangun dengan menggunakan teknologi AJAX, menawarkan solusi untuk mengatasi permasalahan pada sisi klien, dimana permasalahan itu adalah lamanya waktu yang dibutuhkan untuk dapat mengakses informasi atau layanan yang diberikan oleh aplikasi retail komputer berbasis web kepada pelanggannya. Dengan menggunakan teknologi AJAX diharapkan

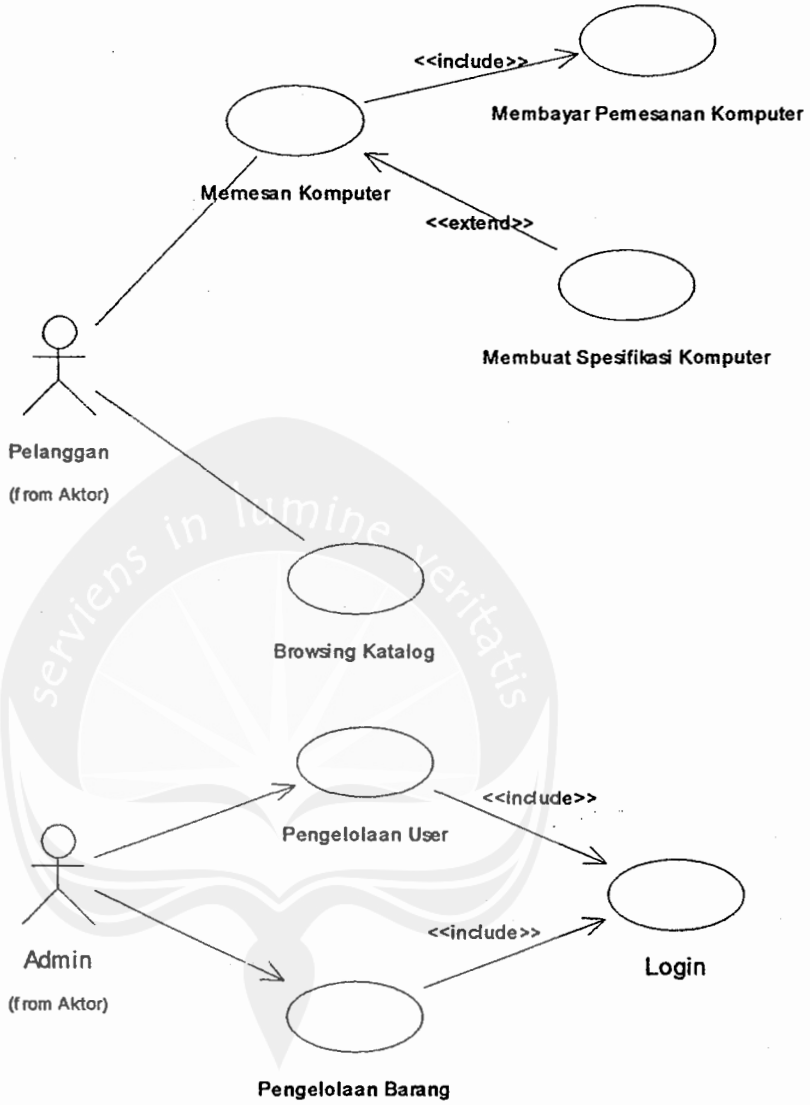
lamanya waktu yang dibutuhkan untuk dapat mengakses informasi atau layanan yang diberikan oleh aplikasi retail komputer berbasis web kepada pelanggannya dapat dikurangi. Sehingga pelanggan tidak stres lagi, dan menjadi lebih betah untuk mencari informasi atau memanfaatkan layanan yang diberikan oleh retail komputer tersebut.

Berikut ini arsitektur sistem ARKAJAX berbasis web yang akan dijalankan melalui koneksi internet :



Gambar 2.1.1. Gambar Arsitektur Sistem ARKAJAX

2.2. Kebutuhan Fungsionalitas Perangkat Lunak



Gambar 2.1 Use Case Diagram untuk ARKAJAX

2.2.1. Use Case : Browsing Katalog

Use case ini digunakan oleh pelanggan untuk mencari informasi komputer yang ingin dipesan. Dengan use case ini pelanggan bisa memperoleh informasi mengenai spesifikasi komputer, harga komputer, serta informasi mengenai fitur-fitur yang yang terdapat dalam komputer.

Lihat : **Spesifikasi Use Case : Browsing Katalog (UC-Brow-ARK01)**

2.2.2. Use Case : Memesan Komputer

Use case ini digunakan oleh pelanggan untuk memesan komputer. Untuk dapat melakukan pemesanan komputer pelanggan harus memilih terlebih dahulu menentukan komputer yang akan dipesan. Pelanggan juga harus mengisikan data yang digunakan untuk melakukan pengiriman barang, misalnya : nama pelanggan, alamat, dan nomor telepon yang dapat dihubungi.

Lihat : **Spesifikasi Use Case : Memesan Komputer (UC-Psn-ARK02)**

2.2.3. Use Case : Membayar Pemesanan Komputer

Use case ini digunakan oleh pelanggan untuk membayar komputer yang dipesan. Pembayaran dilakukan dilakukan via kartu kredit, dimana pelanggan harus memasukkan nomor kartu kredit, tanggal kadaluarsa kartu kredit, dan jumlah uang yang dikeluarkan untuk memesan komputer.

Lihat : **Spesifikasi Use Case : Membayar Pemesanan Komputer (UC-Byr-ARK03)**

2.2.4. Use Case : Menentukan Spesifikasi Komputer

Use case ini digunakan oleh pelanggan untuk menentukan spesifikasi komputer yang dikehendakinya. Use case ini bersifat extend sehingga use case ini dijalankan jika pelanggan ingin memesan komputer yang spesifikasinya ditentukan oleh pelanggan sendiri. Pelanggan dapat menentukan sendiri prosesor, vga card, RAM, motherboard, harddisk, optical drive, FDD, modem, LAN card, mouse, monitor, keyboard, speaker, stabilizer/UPS, Casing.

Lihat : **Spesifikasi Use Case : Menentukan Spesifikasi Komputer (UC-Speckom-ARK04)**

2.2.5. Use Case : Pengelolaan User

Use case ini digunakan oleh Administrator untuk melakukan pengelolaan data pelanggan. Pengelolaan tersebut meliputi input data pelanggan, show data pelanggan, edit data pelanggan, delete data pelanggan. Untuk mengelola data pelanggan administrator harus melewati use case login terlebih dahulu

Lihat : **Spesifikasi Use Case : Pengelolaan User (UC-User-ARK05)**

2.2.6. Use Case : Pengelolaan Barang

Use case ini digunakan oleh Administrator untuk melakukan pengelolaan data barang. Pengelolaan tersebut meliputi input data barang, show data barang, edit data barang, delete data barang. Untuk mengelola data barang administrator harus melewati use case login terlebih dahulu

Lihat : **Spesifikasi Use Case : Pengelolaan Barang (UC- Barang -ARK06)**

2.2.7. Use Case : Login

Use case login digunakan untuk melakukan autentifikasi terhadap user yang akan menjalankan use case pengelolaan user dan use case pengelolaan barang. Jika user tersebut adalah administrator maka user tersebut diijinkan untuk menjalankan use case pengelolaan user dan use case pengelolaan barang, jika bukan administrator maka user tersebut tidak diijinkan untuk menjalankan use case pengelolaan user dan use case pengelolaan barang

Lihat : **Spesifikasi Use Case : Pengelolaan Login (UC- Login - ARK06)**

2.3. Kebutuhan Non Fungsionalitas Perangkat Lunak

Kebutuhan non fungsionalitas perangkat lunak adalah sebagai berikut :

- Aplikasi retail komputer berbasis web dengan menggunakan teknologi AJAX ini dapat menangani penjualan komputer secara online, jika telah terhubung dengan internet.
- Untuk melakukan pembayaran produk yang dipesan digunakan kartu kredit, untuk itu diperlukan pelanggan yang memiliki kartu kredit yang belum kadaluarsa.
- Aplikasi retail komputer berbasis web mampu memberikan informasi tentang pc, server atau notebook secara lengkap yang terdapat pada aplikasi retail komputer berbasis web kepada pelanggan
- Aplikasi retail komputer berbasis web ini menawarkan komputer yang sesuai dengan keinginan pelanggan, dimana pelanggan dapat menentukan sendiri periferal yang terdapat dalam komputer
- Setiap terjadi transaksi aplikasi retail komputer berbasis web ini melakukan verifikasi kartu kredit pelanggan, untuk mengecek apakah kartu kredit tersebut masih dapat digunakan atau tidak.

2.4. Karakteristik Aktor

2.4.1. Pelanggan

Pelanggan adalah target user yang akan dijadikan sebagai konsumen dari produk yang ditawarkan oleh retail komputer. Seorang pelanggan dapat melakukan browsing untuk mencari informasi mengenai produk yang akan dibelinya. Pelanggan juga dapat melakukan pembelian produk.

Untuk dapat melakukan pembelian komputer pelanggan harus memiliki kartu kredit yang belum kadaluarsa. Tidak ada batasan yang spesifik mengenai pelanggan. Yang penting dia adalah oarang yang berminat untuk membeli produk yang

ditawarkan oleh retail komputer dan memiiki kartu kredit yang akan digunakan untuk melakukan pembelian produk.

2.4.2. Admin

Admin adalah pengelola data user dan data barang yang terdapat dalam sistem ARKAJAX, admin memiliki hak untuk mengakses **Use Case : Pengelolaan User, Pengelolaan Barang, Login**

3. Deskripsi Rinci Kebutuhan

3.1. Spesifikasi Kebutuhan Fungsionalitas

3.1.1. Spesifikasi Use Case : Browsing Katalog

Tabel 3.1 Use case : Browsing katalog

Use Case ID	UC-Brow-ARK01
Use Case Name	Browsing katalog
Use Case Type	-
Priority	-
Actors	Pelanggan
Description	Use case ini digunakan oleh pelanggan untuk melakukan browsing katalog penjualan komputer sehingga pelanggan dapat memperoleh informasi mengenai komputer yang dijual oleh retail komputer ini.
Preconditions	-
Basic Path	1. Sistem menampilkan antar muka menu katalog 2. Aktor memilih jenis katalog yang akan dilihat 3. Sistem menampilkan katalog yang dipilih oleh pelanggan
Alternative Paths	-
Postconditions	Aktor memperoleh informasi komputer

Exception Paths	-
Extends	-
Includes	-
Business Rules	-

3.1.2. Spesifikasi Use Case : Memesan Komputer

Tabel 3.2 Use case : Memesan komputer

Use Case ID	UC-Psn-ARK02
Use Case Name	Memesan komputer
Use Case Type	-
Priority	-
Actors	Pelanggan
Description	Use case ini digunakan oleh pelanggan untuk Memesan komputer. Pemesanan komputer dilakukan dengan cara mengisi formulir pemesanan yang disediakan oleh sistem
Preconditions	-
Basic Path	1. Sistem menampilkan antarmuka pemesanan komputer 2. Aktor memasukkan data pemesanan komputer 3. Sistem mengecek kelengkapan formulir pemesanan komputer 4. Sistem mengkonfirmasi pemesanan komputer
Alternative Paths	-
Postconditions	Aktor masuk ke halaman pembayaran pemesanan komputer
Exception Paths	**1. Formulir pemesanan tidak lengkap (Tabel 3.2 Langkah 3)**

	a. Sistem menampilkan pesan bahwa formulir pemesaan belum lengkap b. Sistem menampilkan antarmuka pemesanan komputer
Extends	**Spesifikasi Use Case : Menentukan Spesifikasi Komputer (UC-Speckom-ARK04)**
Includes	**Spesifikasi Use Case : Membayar Pemesanan Komputer (UC-Byr-ARK03)**
Business Rules	-

3.1.3. Spesifikasi Use Case : Membayar Pemesanan Komputer

Tabel 3.3 Use case : Membayar pemesanan komputer

Use Case ID	UC-Byr-ARK03
Use Case Name	Membayar pemesanan komputer
Use Case Type	-
Priority	-
Actors	Pelanggan
Description	Use case ini digunakan oleh pelanggan untuk membayar pemesanan komputer. Pembayaran dilakukan dengan menggunakan kartu kredit yang masih berlaku. Pembayaran pesanan komputer dilakukan dengan mengisikan formulir pembayaran yang disediakan oleh sistem
Preconditions	Aktor telah mengisi formulir pemesanan komputer
Basic Path	1. Sistem menampilkan formulir pembayaran 2. Aktor mengisi formulir pembayaran 3. Sistem mengecek kelengkapan formulir pembayaran 4. Sistem memvalidasi kartu kredit 5. Sistem mengkonfirmasi pembayaran pemesanan

	komputer
Alternative Paths	-
Postconditions	-
Exception Paths	**1. Formulir pembayaran tidak lengkap (tabel 3.3 langkah 3)** a. Sistem menampilkan pesan bahwa formulir pembayaran belum lengkap b. Sistem menampilkan antarmuka pembayaran komputer **2. Kartu kredit sudah tidak berlaku (tabel 3.3 langkah 4)** a. Sistem menampilkan pesan bahwa kartu kredit sudah tidak berlaku / kadaluarsa b. Sistem menanyakan pada aktor apakah akan menggunakan kartu kredit yang lain i. Jika menggunakan kartu kredit yang lain maka kembali ke **tabel 3.3 langkah 1** ii. Jika tidak maka aktor keluar dari sistem pembayaran
Extends	-
Includes	-
Business Rules	-

3.1.4. Spesifikasi Use Case : Menentukan Spesifikasi Komputer

Tabel 3.4 Use case : Menentukan spesifikasi komputer

Use Case ID	UC-Speckom-ARK04
Use Case Name	Menentukan spesifikasi komputer
Use Case Type	-
Priority	-
Actors	Pelanggan

Description	Use case ini digunakan oleh pelanggan untutk memesan komputer yang spesifikasinya ditentukan oleh pelanggan sendiri. Pelanggan dapat menentukan sendiri produk yang akan dipesan.
Preconditions	Aktor sudah masuk ke halaman pemesanan komputer
Basic Path	1. Aktor memilih menu untuk menentukan spesifikasi komputer 2. Sistem menampilkan formulir untuk menentukan spesifikasi komputer 3. Aktor memasukkan spesifikasi komputer yang akan dipesan 4. Sistem mengecek kelengkapan formulir 5. Sistem mengkonfirmasi penentuan spesifikasi komputer
Alternative Paths	-
Postconditions	Aktor masuk ke halaman pemesanan komputer
Exception Paths	1. **Formulir penentuan spesifikasi komputer tidak lengkap (tabel 3.4 langkah 4)** a. Sistem menampilkan pesan bahwa formulir belum lengkap b. Sistem menampilkan antarmuka spesifikasi perangkat komputer yang akan dipesan
Extends	-
Includes	-
Business Rules	-

3.1.5. Spesifikasi Use Case : Pengelolaan User

Tabel 3.5 Use case : Pengelolaan User

Use Case ID	UC-User-ARK05
Use Case Name	Pengelolaan User

Use Case Type	-
Priority	-
Actors	Admin
Description	Use case ini digunakan oleh Administrator untuk melakukan pengelolaan data pelanggan. Pengelolaan tersebut meliputi input data pelanggan, show data pelanggan, edit data pelanggan, delete data pelanggan
Preconditions	**Use Case : Login** sudah dilakukan dan actor sudah memasuki sistem
Basic Path	1. Sistem menampilkan antarmuka pengelolaan user 2. Aktor memilih jenis pengelolaan 3. Aktor memasukan inputan yang diminta oleh sistem 4. Sistem mengeksekusi permintaan aktor 5. Sistem menampilkan hasil eksekusi
Alternative Paths	-
Postconditions	-
Exception Paths	-
Extends	-
Includes	**Use Case : Login (UC-Login-ARK07)**
Business Rules	-

3.1.6. Spesifikasi Use Case : Pengelolaan Barang

Tabel 3.6 Use case : Pengelolaan Barang

Use Case ID	UC-Barang-ARK06
Use Case Name	Pengelolaan Barang
Use Case Type	-

Priority	-
Actors	Admin
Description	Use case ini digunakan oleh Administrator untuk melakukan pengelolaan data barang. Pengelolaan tersebut meliputi input data barang, show data barang, edit data barang, delete data barang
Preconditions	**Use Case : Login** sudah dilakukan dan actor sudah memasuki sistem
Basic Path	1. Sistem menampilkan antarmuka pengelolaan barang 2. Aktor memilih jenis pengelolaan barang 3. Aktor memasukan inputan yang diminta oleh sistem 4. Sistem mengeksekusi permintaan aktor 5. Sistem menampilkan hasil eksekusi
Alternative Paths	-
Postconditions	-
Exception Paths	-
Extends	-
Includes	**Use Case : Login (UC-Login-ARK07)**
Business Rules	-

3.1.7. Spesifikasi Use Case : Login

Tabel 3.7 Use case : Login

Use Case ID	UC-Barang-ARK07
Use Case Name	Login
Use Case Type	-
Priority	-
Actors	Admin

Description	Use case ini digunakan oleh Administrator untuk memperoleh akses ke sistem administrator.
Preconditions	-
Basic Path	1. Menampilkan antarmuka Login 2. Aktor memasukkan username dan password 3. Sistem memeriksa username dan password aktor 4. Sistem memberikan akses ke aktor
Alternative Paths	-
Postconditions	Admin masuk ke dalam sistem administrator dan dapat menggunakan fungsi-fungsi dalam sistem
Exception Paths	1. **Username dan password tidak sesuai (table 3.7. langkah 2)** a. Sistem menampilkan peringatan bahwa username dan password tidak sesuai
Extends	-
Includes	-
Business Rules	-

3.2. Spesifikasi Kebutuhan Non Fungsionalitas

3.2.1. Kebutuhan Antarmuka Eksternal

Kebutuhan antar muka eksternal pada aplikasi retail komputer berbasis web mencakup kebutuhan antarmuka pemakai, antarmuka perangkat keras dan antarmuka perangkat lunak.

3.2.2. Kebutuhan Antarmuka Pemakai

Pemakai berinteraksi langsung dengan aplikasi retail komputer berbasis web. Piranti masukan yang digunakan untuk memasukkan data masukan adalah *keyboard* dan *mouse*. Sedangkan keluaran dari sistem berupa data-data yang disimpan

dalam basis data dan dalam bentuk file serta akan ditampilkan langsung ke layar monitor.

3.2.3. Kebutuhan Antarmuka Perangkat Keras

Antarmuka perangkat keras yang dibutuhkan dalam penggunaan perangkat lunak aplikasi retail komputer berbasis web adalah:

1. Development
   - *PC Compatible AMD Barton 2500+ atau Pentium 4*
   - *Keyboard*
   - *Mouse*
   - *Printer*
   - *RAM 512 MB*
2. Client/Server
   a. Server
      - *PC Compatible Pentium 4*
      - *Keyboard*
      - *Mouse*
      - *RAM 512 MB*
      - *Layar monitor yang memiliki warna 16-bit dengan resolusi 1024 x 768*

   b. Client
      - *PC Compatible Pentium 3*
      - *Keyboard*
      - *Mouse*
      - *Network Interface Card*
      - *Layar monitor yang memiliki warna 16-bit dengan resolusi 1024 x 768*
      - *RAM 128 MB*

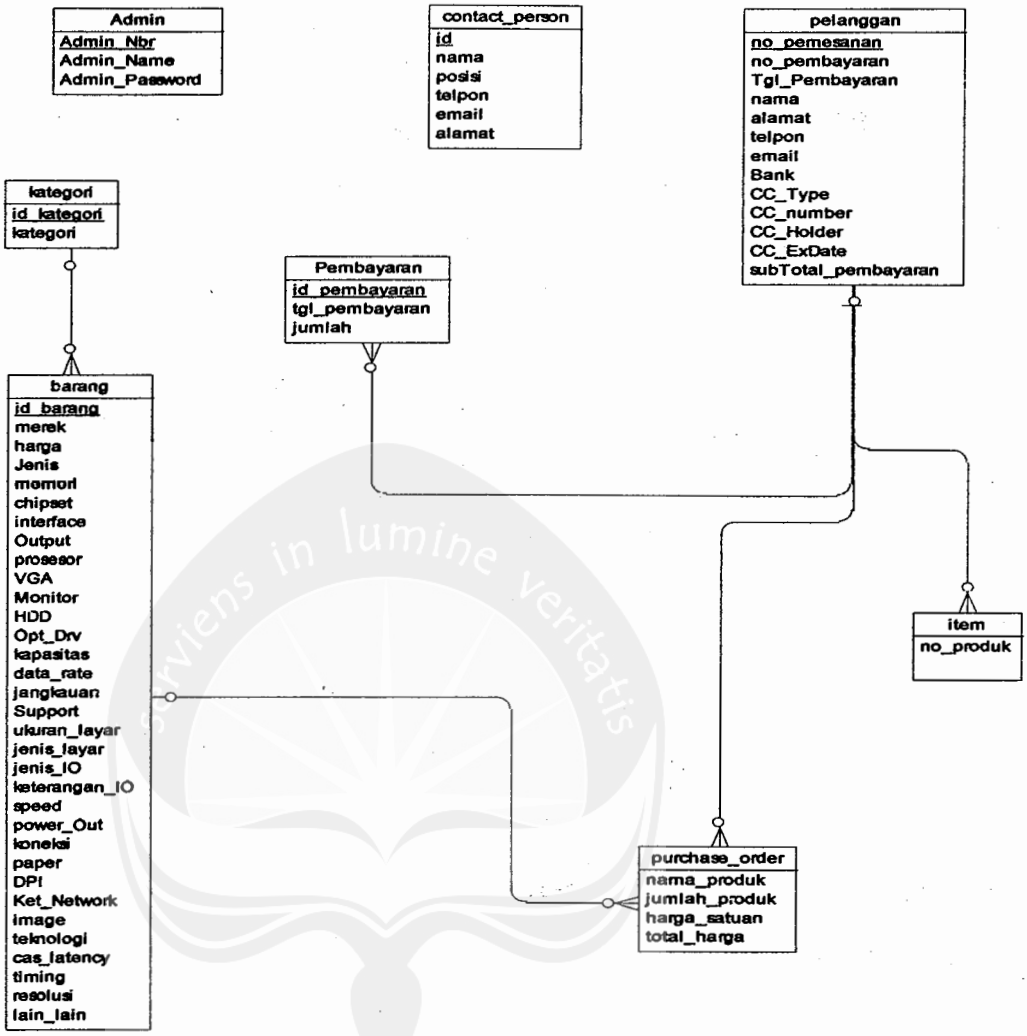
3.2.4. Kebutuhan Antarmuka Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk mendukung berjalannya perangkat lunak aplikasi retail komputer berbasis web adalah:

- Nama : *Windows XP*
  Sumber : *Microsoft*
  Fungsi : Sistem Operasi Komputer
- Nama : *Visual Basic .NET*
  Sumber : *Microsoft*
  Fungsi : *Tool* perancang antarmuka aplikasi
- Nama : *SQL Server 2000*
  Sumber : *Microsoft*
  Fungsi : *Database Management System*
- Nama : *Internet Explorer 6.0*
  Sumber : *Microsoft*
  Fungsi : *Web Browser*
- Nama : *Javascript*
  Sumber : *Netscape*
  Fungsi : *Client-side script dan perancang AJAX*

3.2.5. Persistent Data

3.2.5.1. Basis Data



Gambar 3.1 Entity Relationship Diagram sistem ARKAJAX

3.2.5.2. Kamus Data

i. Tabel 1: Tabel Pelanggan

Atribut	Tipe Data	Ukuran	Deskripsi
No_pemesanan	Numeric	10	Primary key pelanggan
Nama	Char	20	Nama pelanggan
Alamat	Char	100	Alamat pelanggan yang digunakan sebagai tujuan

			digunakan sebagai tujuan pengiriman barang
Telepon	Numeric	10	Nomor telpon pelanggan yang bisa dihubungi
Email	Char	20	Email pelanggan
Kartu Kredit	Numeric	11	Kartu kredit yang dimiliki pelanggan

ii. Tabel 2: Tabel Barang

Atribut	Tipe data	Ukuran	Deskripsi
Kode_Barang	Numeric	10	Primary key barang
Nama	Char	20	Nama barang
Jenis	Char	20	Jenis barang
Harga	Money		Harga barang
Kategori_barang	Char	10	Kategori barang

iii. Tabel 3: Tabel Memesan

Atribut	Tipe data	Ukuran	Deskripsi
No_pemesanan	Numeric	10	Primary key pelanggan
Kode_Barang	Numeric	10	Primary key barang
Jumlah_barang	Numeric	10	Jumlah barang yang dipesan oleh pelanggan
Total_harga	Money		Total harga pemesanan

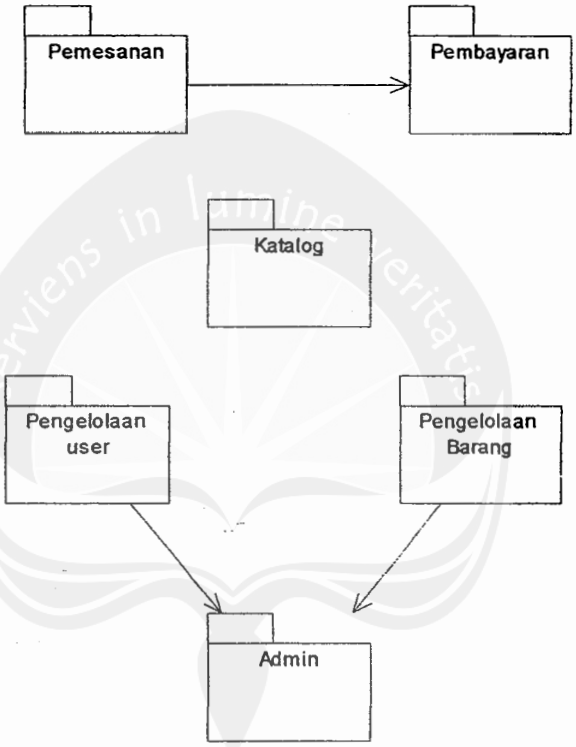
iv. Table 4: Table Admin

Atribut	Tipe data	Ukuran	Deskripsi
No_Admin	Char	10	Primary key Admin
Nama_Admin	Char	50	Nama administrator
Password	Char	50	Password admin

4. Realisasi Use Case

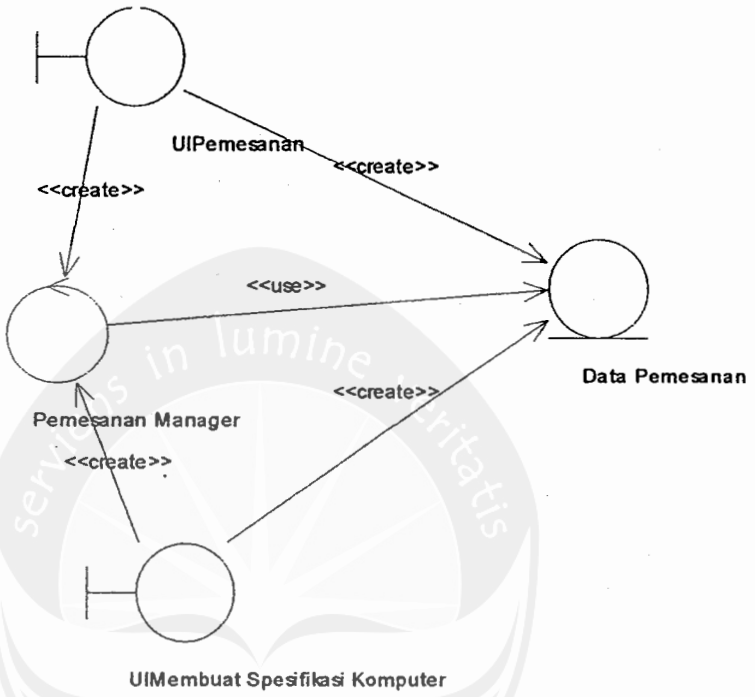
4.1. Static Structure Diagram

4.1.1. Analisis Class Diagram : Package Dependencies



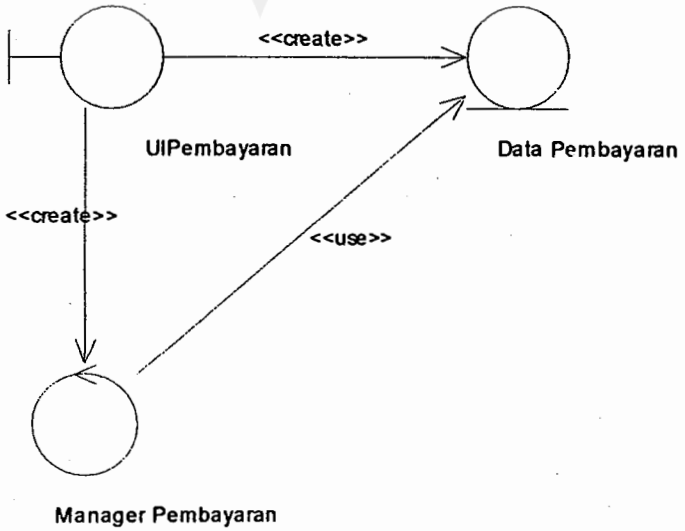
Gambar 4.1.1 Analysis Class Diagram : Package Dependencies

4.1.2. Analisis Class Diagram : Package Pemesanan



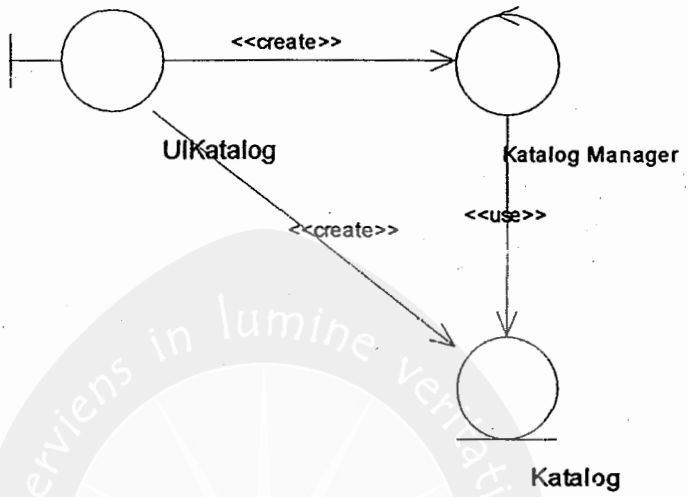
Gambar 4.1.2 Analysis Class Diagram : Package Pemesanan

4.1.3. Analisis Class Diagram : Package Pembayaran



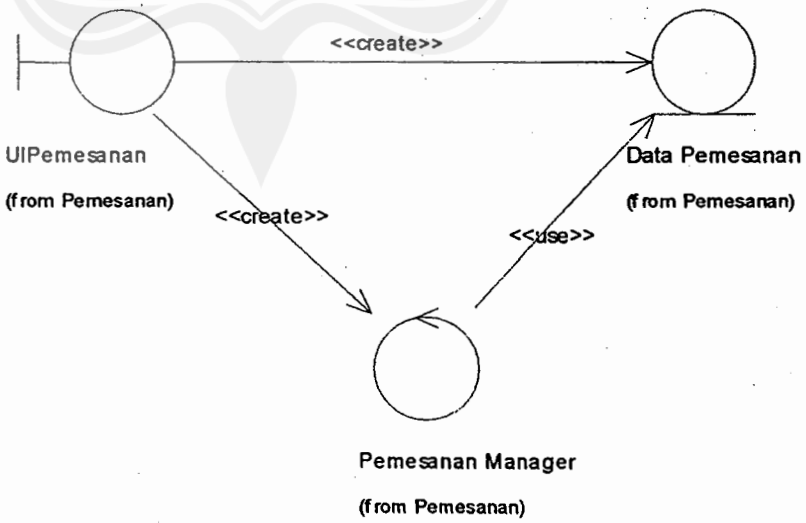
Gambar 4.1.3 Analysis Class Diagram : Package Pembayaran

4.1.4. Analisis Class Diagram : Use Case Package Katalog



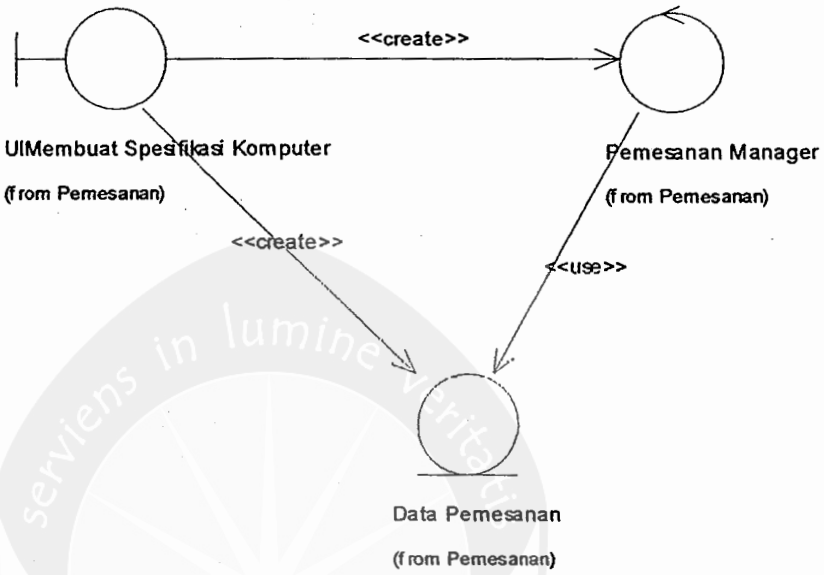
Gambar 4.1.4 Analysis Class Diagram : Package Katalog

4.1.5. Analisis Class Diagram : Use Case Memesan Komputer



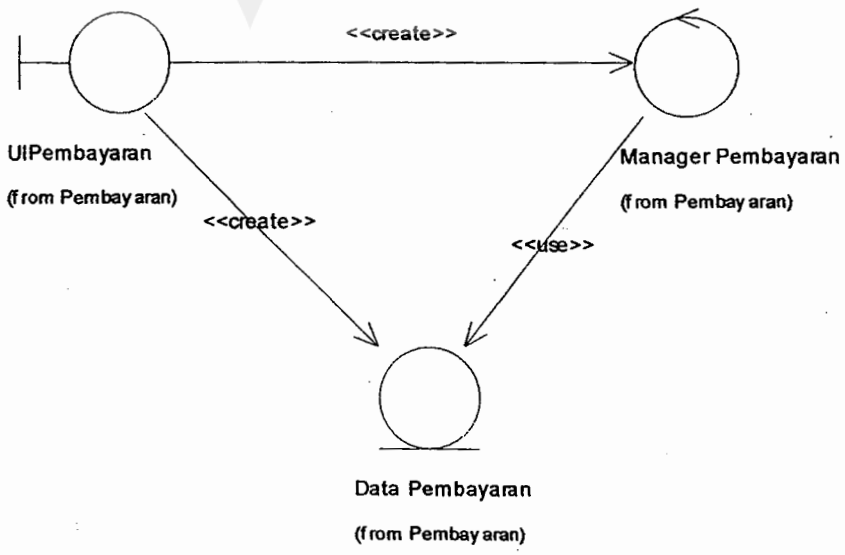
Gambar 4.1.5 Analysis Class Diagram : Use Case Memesan Komputer

4.1.6. Analisis Class Diagram : Use Case Menentukan Spesifikasi Komputer



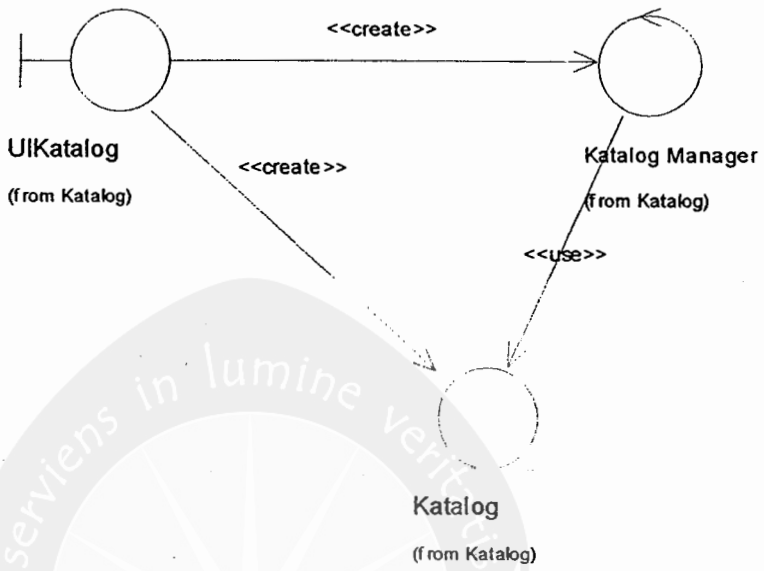
Gambar 4.1.6 Analysis Class Diagram : Use Case Menentukan Spesifikasi Komputer

4.1.7. Analisis Class Diagram : Use Case Membayar Pemesanan Komputer



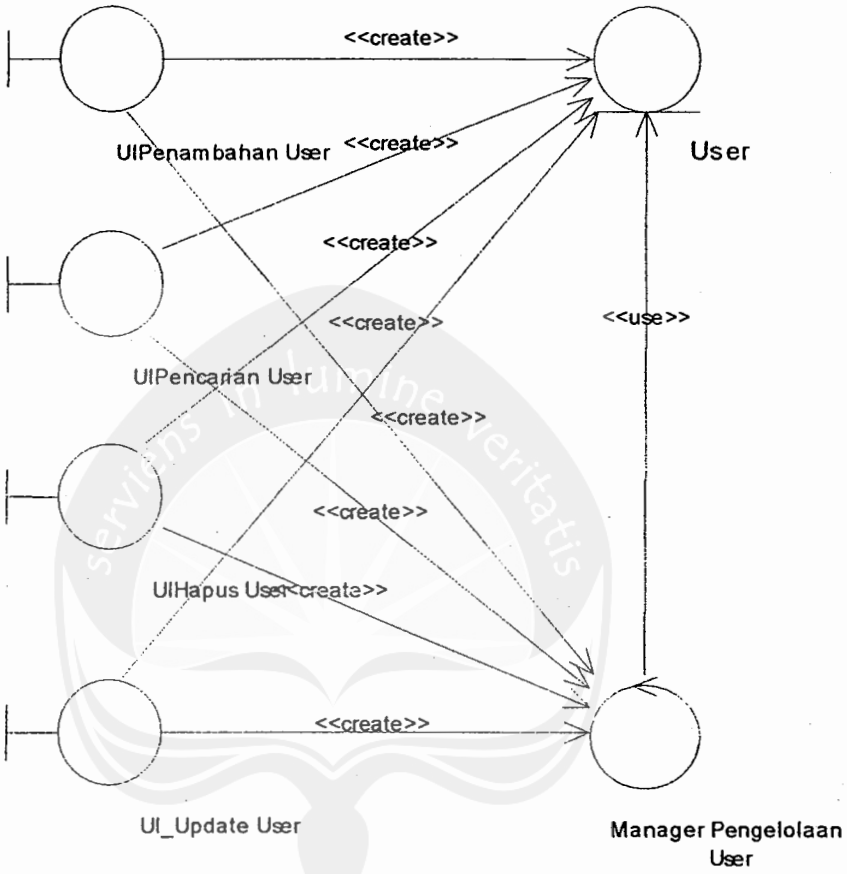
Gambar 4.1.7 Analysis Class Diagram : Use Case Membayar Pemesanan Komputer

4.1.8. Analisis Class Diagram : Use Case Browsing Katalog



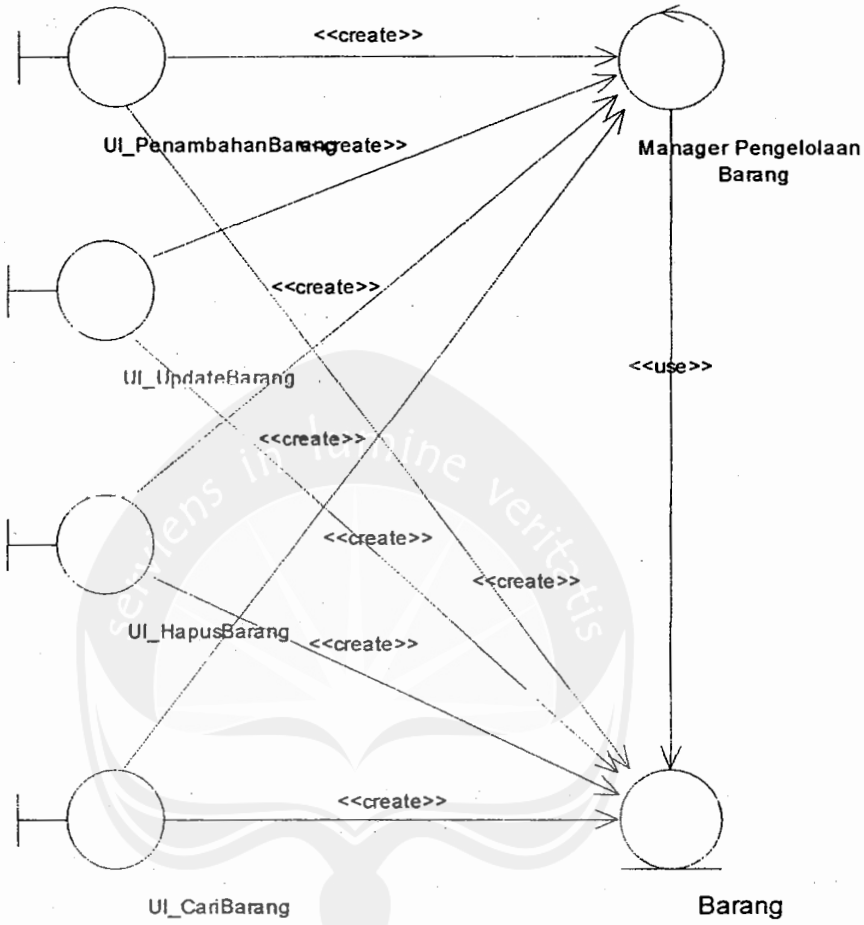
Gambar 4.1.8 Analysis Class Diagram : Use Case Membayar Pemesanan Komputer

4.1.9. Analisis Class Diagram : Use Case Pengelolaan User



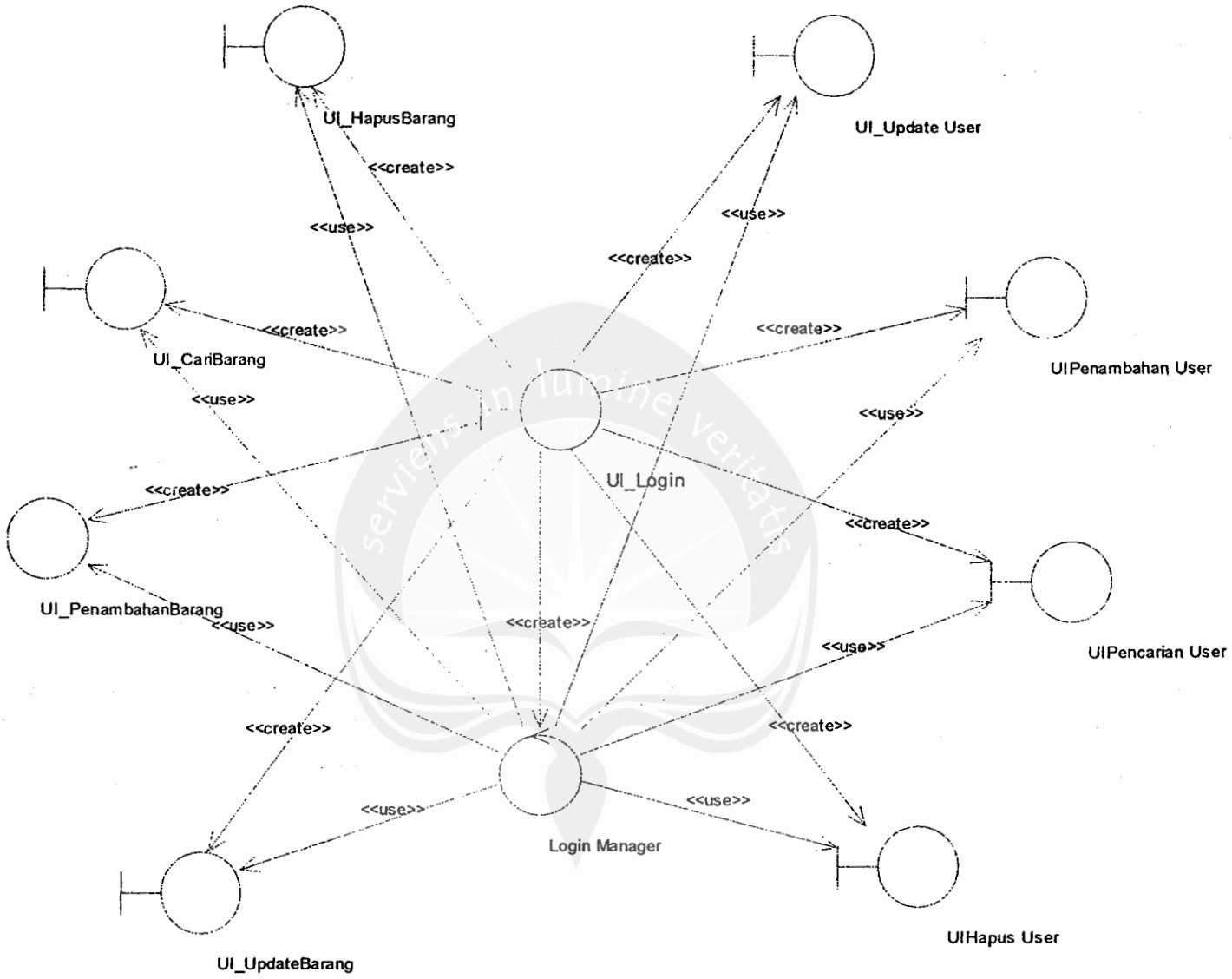
Gambar 4.1.9 Analysis Class Diagram : Use Case Pengelolaan User

4.1.10. Analisis Class Diagram : Use Case Pengelolaan Barang



Gambar 4.1.10 Analysis Class Diagram : Use Case Pengelolaan Barang

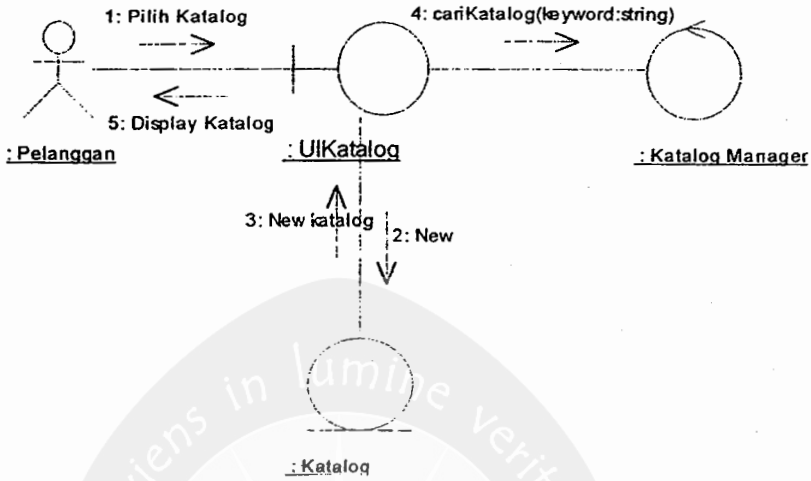
4.1.11. Analisis Class Diagram : Use Case Login



Gambar 4.1.11 Analysis Class Diagram : Use Case Login

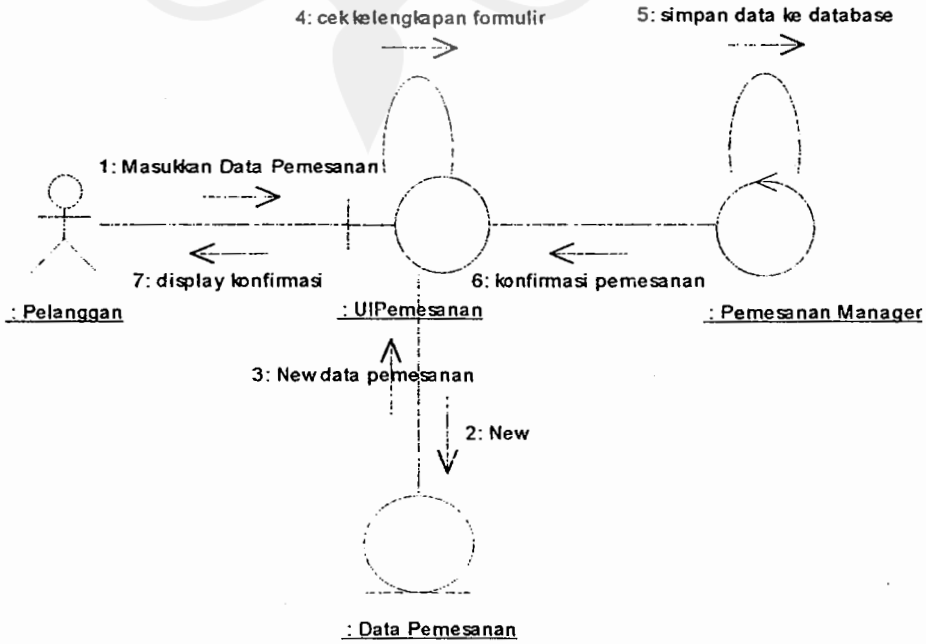
4.2. Interaction Diagram

4.2.1. Analisis Collaboration Diagram : Use Case Browsing Katalog



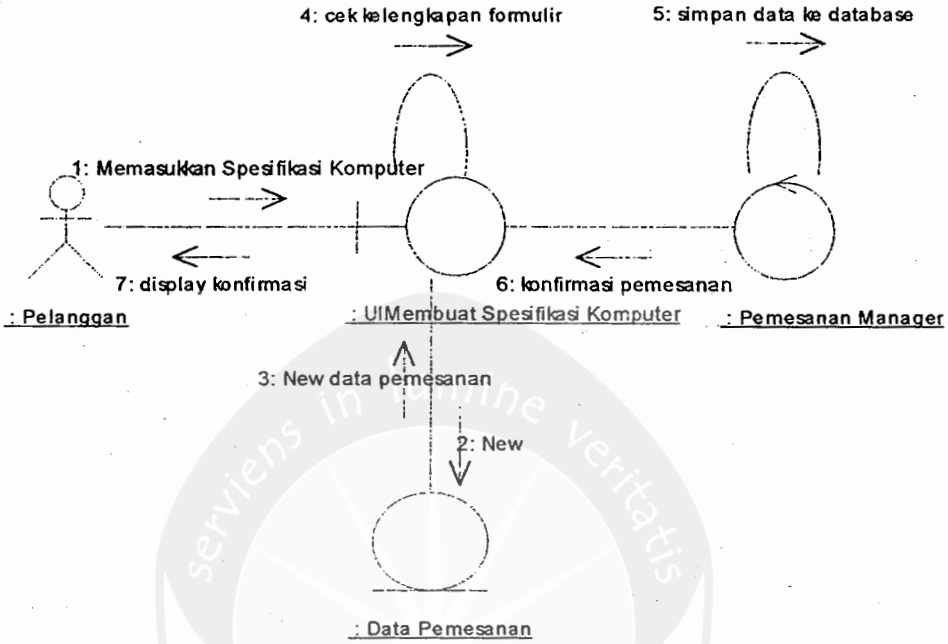
Gambar 4.2.1 Analysis Collaboration Diagram : Use Case Browsing Katalog

4.2.2. Analisis Collaboration Diagram : Use Case Memesan Komputer



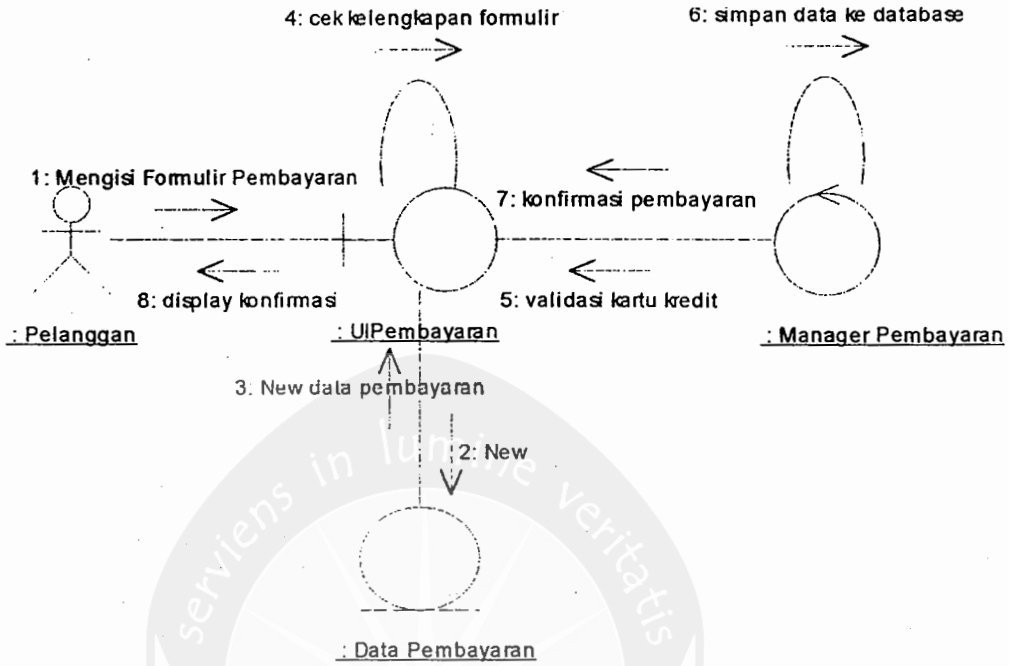
Gambar 4.2.2 Analysis Collaboration Diagram : Use Case Memesan Komputer

4.2.3. Analisis Collaboration Diagram : Use Case Menentukan Spesifikasi Komputer



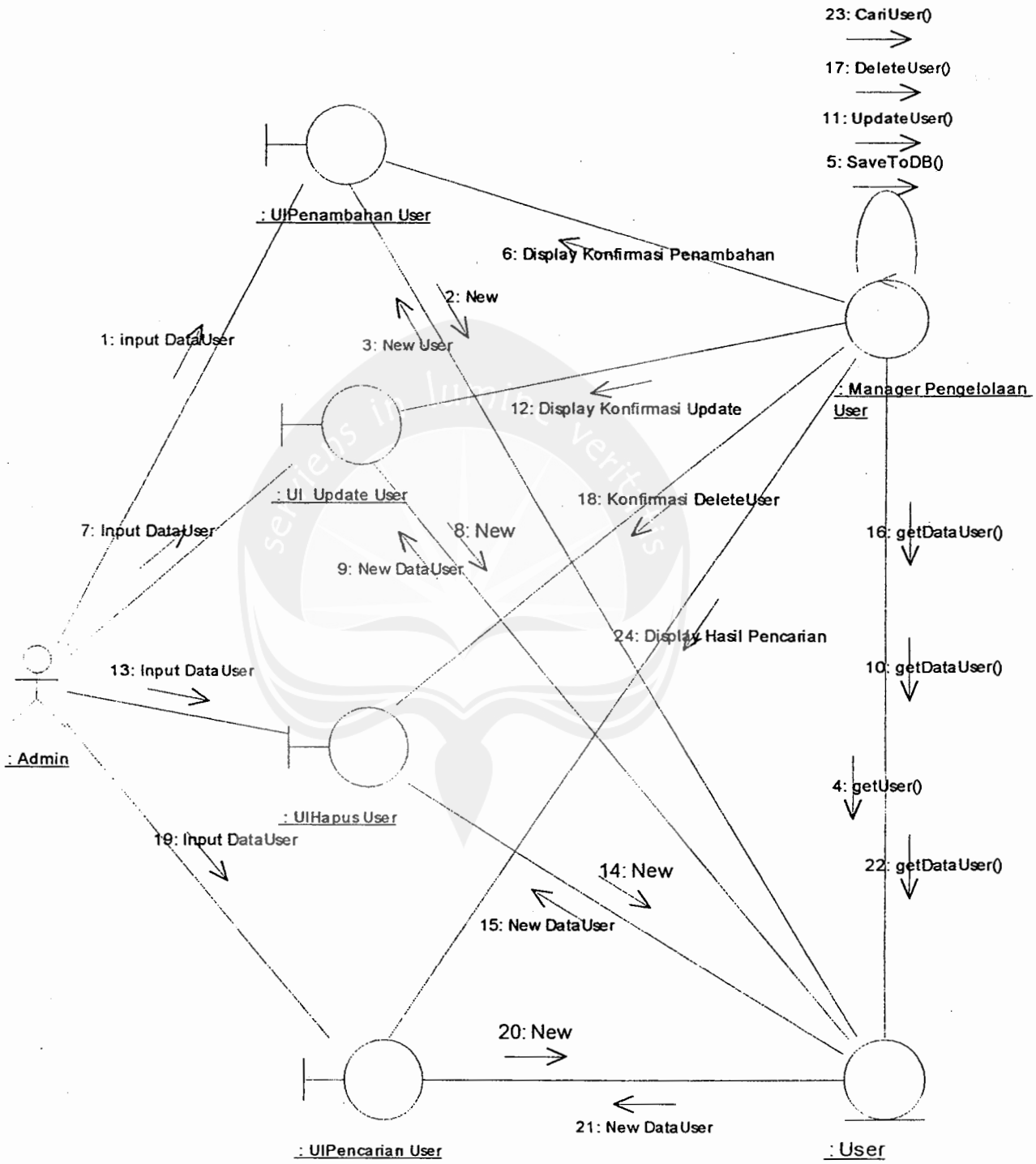
Gambar 4.2.3 Analysis Collaboration Diagram : Use Case Menentukan Spesifikasi Komputer

4.2.4. Analisis Collaboration Diagram : Use Case Membayar Pemesanan Komputer



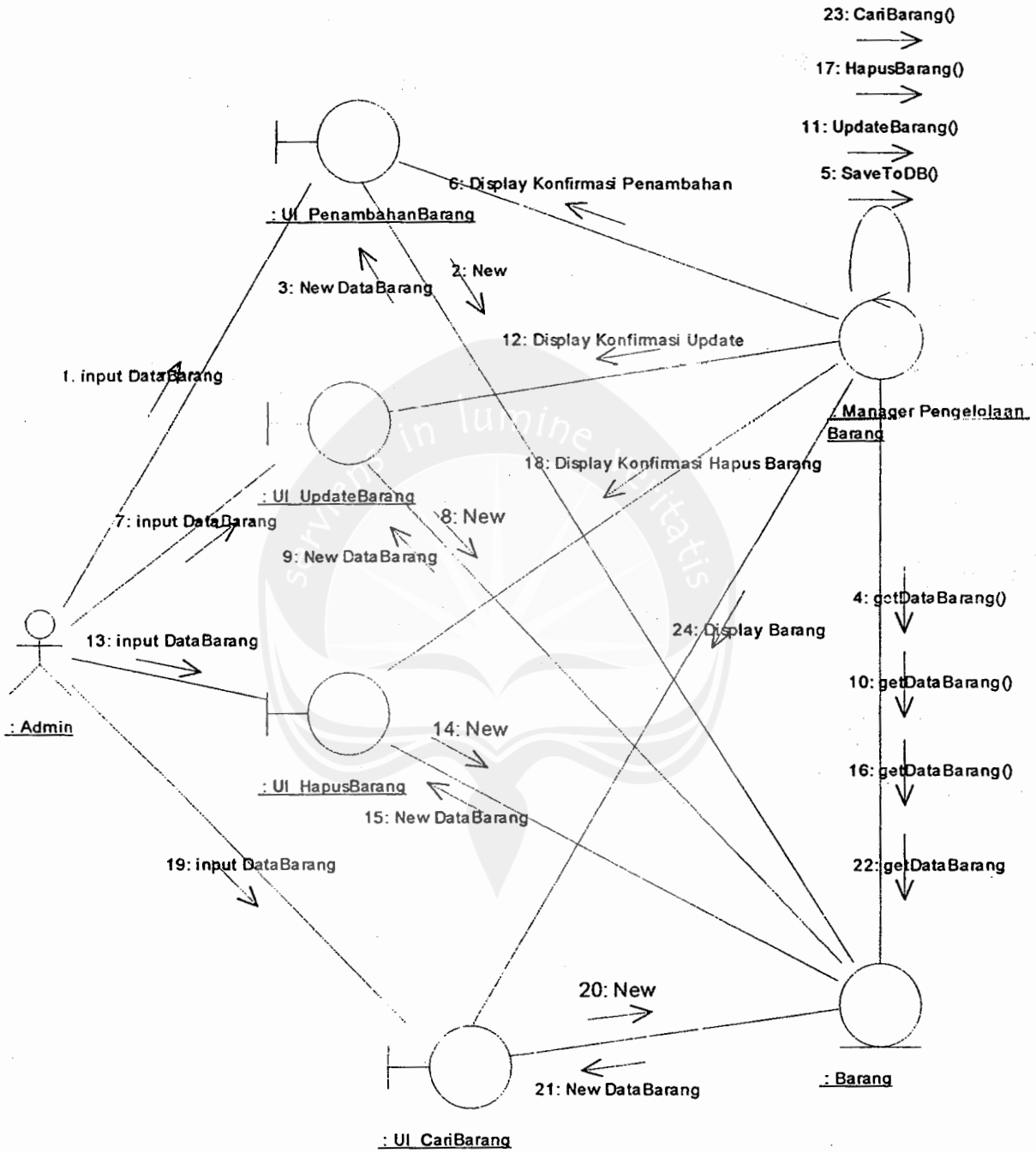
Gambar 4.2.4 Analysis Collaboration Diagram : Use Case Membayar Pemesanan Komputer

4.2.5. Analisis Collaboration Diagram : Use Case Pengelolaan user



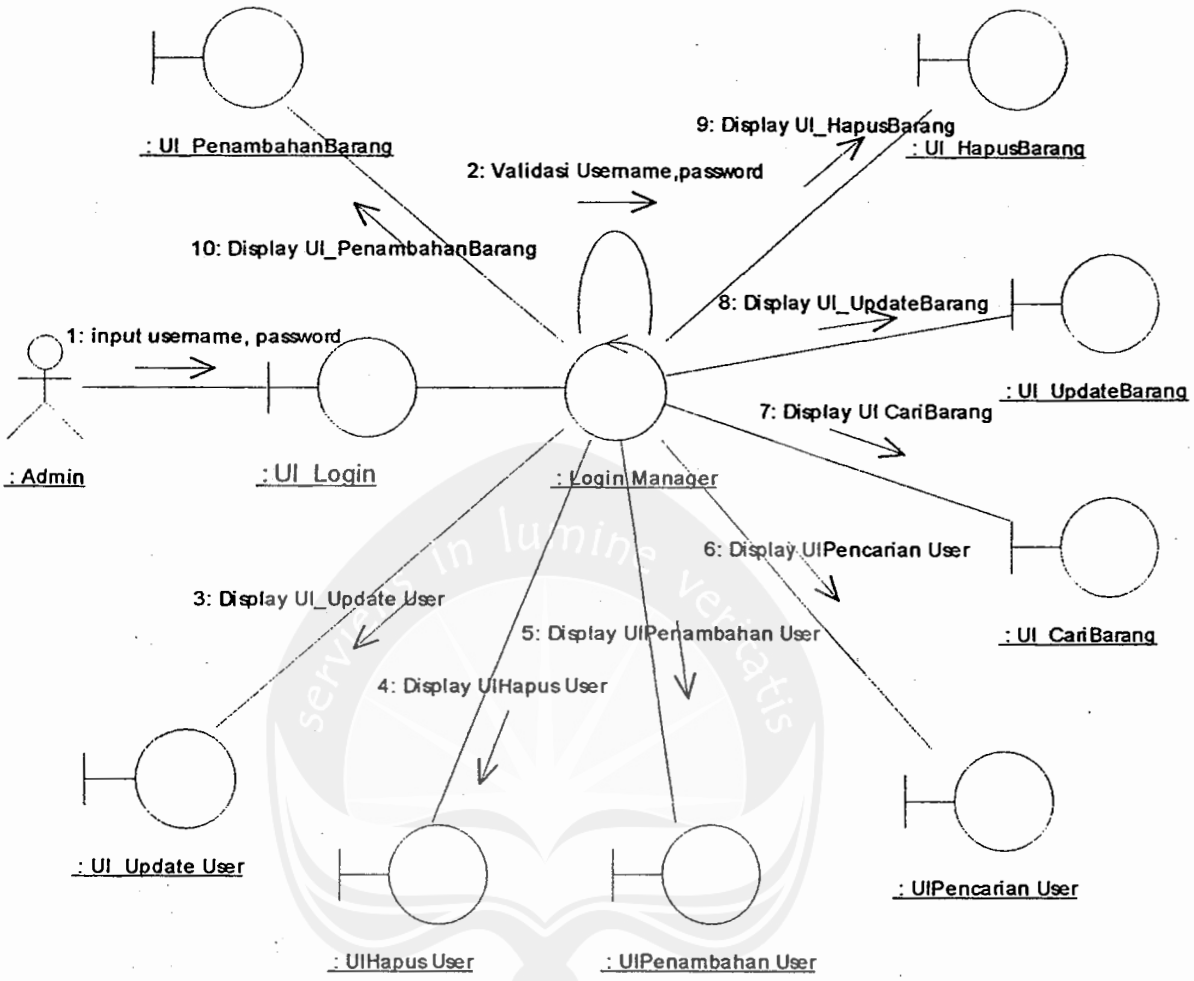
Gambar 4.2.5 Analysis Collaboration Diagram : Use Case Pengelolaan User

4.2.6. Analisis Collaboration Diagram : Use Case Pengelolaan Barang



Gambar 4.2.6 Analysis Collaboration Diagram : Use Case Pengelolaan Barang

4.2.7. Analisis Collaboration Diagram : Use Case Login



Gambar 4.2.7 Analysis Collaboration Diagram : Use Case Login

DPPL

Deskripsi Perancangan Perangkat Lunak

Aplikasi Retail Komputer Berbasis Web Dengan Menggunakan Teknologi AJAX (ARKAJAX)

Bagian dari Tugas Akhir

Disusun oleh:

Kuntoro Haryatmoko (TF/2979)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ATMA JAYA YOGYAKARTA

DAFTAR PERUBAHAN

Revisi	Deskripsi
1	

INDEX TGL	-	A	B	C	D	E	F
Ditulis oleh							
Diperiksa oleh							
Disetujui oleh							

NOTASI DOKUMEN

Notasi yang digunakan dalam dokumen ini adalah sebagai berikut :

- Teks normal ditulis dalam *font* Times New Roman 12 pt, *plain*.
- Teks yang ditulis dalam *font* ***bold*** merupakan teks yang mengacu pada bab, sub-bab, gambar, atau tabel dalam dokumen ini.
- Teks yang ditulis dalam *font* Courier New merupakan teks yang mengacu pada model, diagram, atau file yang disebutkan dalam dokumen ini.

DAFTAR ISI

Aplikasi Retail Komputer Berbasis Web Dengan Menggunakan Teknologi AJAX (ARKAJAX)

1. Pendahuluan

1.1. Tujuan

Dokumen DPPL ini dibuat untuk menyediakan deskripsi lengkap mengenai desain perangkat lunak ARKAJAX, yang dibuat untuk melakukan penjualan komputer secara *online*. Dokumen ini ditujukan untuk pembuat perangkat lunak, dan orang lain yang tertarik untuk mengembangkan perangkat lunak ini

1.2. Lingkup Dokumen

Dokumen DPPL ini menyediakan deskripsi yang lengkap untuk perancangan sistem ARKAJAX. Perancangan ini merupakan arsitektur sistem yang menjelaskan perancangan class/modul, detail operasi apa saja yang dilakukan oleh masing-masing class/modul, *layout database*.

1.3. Definisi Istilah dan Singkatan

Definisi untuk istilah dan singkatan yang digunakan dalam dokumen ini dapat mengacu pada **Apendiks A: Daftar Istilah dan Singkatan**.

1.4. Referensi

Referensi yang digunakan pada perangkat lunak tersebut adalah:

1. Timotius Pamungkas, *Deskripsi Perancangan Perangkat Lunak JPAM Gradding Tool*, Program Studi Teknik Informatika – UAJY
2. Martin Fowler, *UML Distilled*, Andi Offset Yogyakarta, 2005

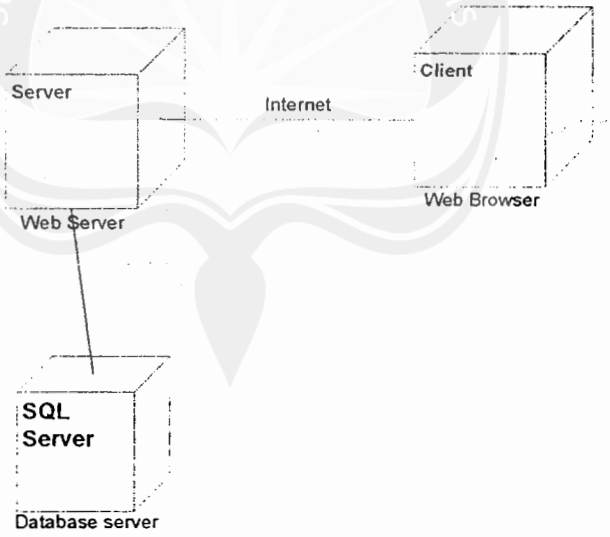
1.5. Deskripsi Umum Dokumen

Dokumen ini terdiri atas empat 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 dalam *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

Deployment ini dibuat untuk menunjukkan semua *node* (lihat **Apendiks A: Daftar Istilah dan Singkatan**) pada sistem, hubungan diantara mereka dan proses yang dijalankan di masing-masing *node*.



Gambar 2.1 Deployment Diagram ARKAJAX

2.1.1. Node Client

Client merupakan komputer yang digunakan oleh pelanggan untuk melakukan pemesanan produk atau melakukan *browsing* produk komputer yang dijual di *retail* komputer, proses yang digunakan didalamnya adalah:

1. *Web browser*, digunakan untuk melakukan browsing produk yang terdapat di *retail* komputer, dan digunakan untuk melakukan pemesanan produk yang terdapat di retail komputer

2.1.2. Node Server

Server merupakan komputer yang digunakan untuk melayani data pemesanan pada client dan menangani pengelolaan data. Proses yang terdapat di dalamnya adalah:

1. Aplikasi *server*, digunakan untuk melayani data pemesanan yang dibutuhkan oleh *client*

2.1.3. Node Database Server

Database server merupakan komputer yang digunakan untuk menyimpan data pemesanan

1. SQL Server 2000, digunakan untuk menyimpan data pemesanan, dan pembayaran produk

2.2. Design Class

2.2.1. Pengantar

Tipe *data* hanya akan ditulis tanpa *package* dan diawali dengan huruf non-kapital

Stereotype yang digunakan dalam *design class* adalah:

- *<<boundary>>*

  Boundary class merupakan *class* yang digunakan untuk menghubungkan sistem dengan *user* diluar sistem

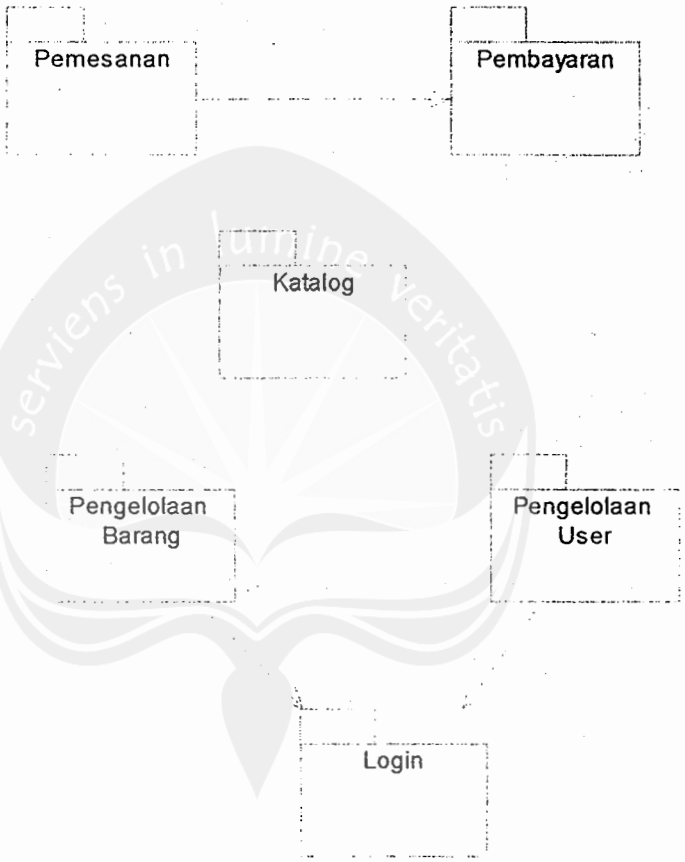
- *<<control>>*

  Class control merupakan *class* yang objeknya melakukan interaksi antar sekelompok objek lain. *Class control* biasanya memiliki karakteristik yang spesifik untuk satu *use case*, dan objek *class* ini biasanya hanya aktif pada realisasi *use case*

- *<<entity>>*

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

2.2.2. Package Dependencies

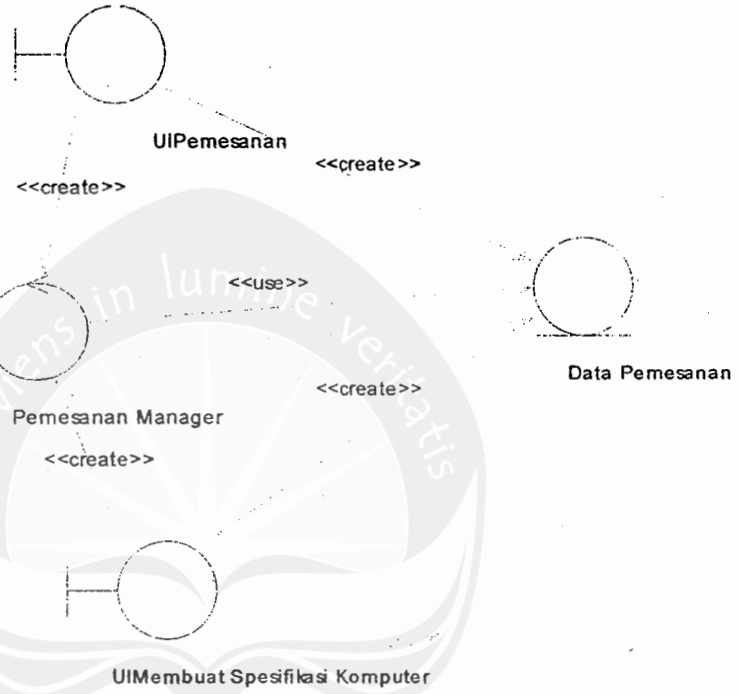


Gambar 2.2 *Deployment Diagram* ARKAJAX

2.2.3. Package Pemesanan

Package ini menyediakan class-class yang digunakan oleh sistem untuk mengelola hal-hal yang berhubungan dengan pemesanan

2.2.3.1. Class Diagram Package Pemesanan



Gambar 2.3 *Class Diagram Package* Pemesanan

2.2.3.2. Class Data Pemesanan

Gambar 2.4 *Class* Data Pemesanan

- **Deskripsi**

  Class yang mempresentasikan data transaksi pemesanan produk

- **Atribut**
  - -no_pemesanan: Integer

merepresentasikan nomor pemesanan

- `- merek_item: String`

  merepresentasikan merek produk yang dipesan

- `-jenis_item: String`

  merepresentasikan jenis *item* yang dipesan

- `-jumlah_item: Integer`

  merepresentasikan jumlah *item* yang dipesan

- `-harga_item: Integer`

  merepresentasikan harga satuan *item* yang dipesan

- `-email_pemesan: String`

  merepresentasikan *email* pemesan

- `-alamat_pemesan: String`

  merepresentasikan alamat pemesan

- `-nama_pemesan: String`

  merepresentasikan nama pemesan

- `-tgl_pemesanan : Date`

  merepresentasikan waktu pemesanan

- `-telpon_pemesan : Integer`

  merepresentasikan nomor telpon pemesanan

- `-kode_item : Integer`

  merepresentasikan kode item yang dipesan

- **Method**

  - `+DataPemesanan()`

    Buat *instance* baru `DataPemesanan` tanpa atribut terdefinisi.

  - `+DataPemesanan(no_pemesanan : Integer, merek_item : String, jenis_item : String, jumlah_item : Integer, harga_item : Double, email_pemesan : String, nama_pemesan : String, alamat_pemesan :`

String, tgl_pemesanan : Date, kode_item : Integer)

Buat *instance* baru DataPemesanan dengan atribut terdefinisi.

Parameters:

no_pemesanan - nomor pemesanan

merek_item – merek produk

jenis_item – jenis produk

jumlah_item – jumlah produk

harga_item – harga produk

email_pemesan – email pelanggan

nama_pemesan – nama pelanggan

alamat_pemesan – alamat pelanggan

tgl_pemesanan – tanggal terjadinya pemesanan

kode_item – kode item yang dipesan

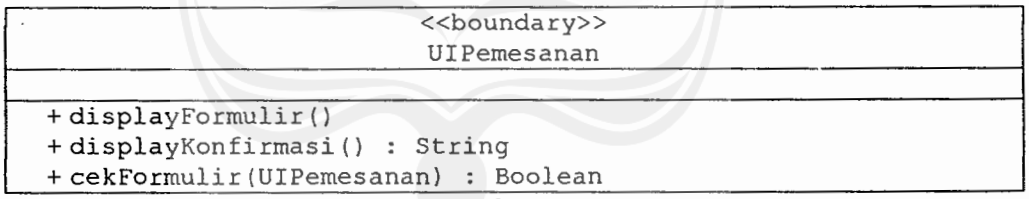
Method-method berikut merupakan *accessor* untuk atribut *class* DataPemesanan :

- +getMerekItem() : String
- +getKodeItem() : Integer
- +getNoPemesanan() : Integer
- +getJenisItem() : String
- +getJumlahItem() : Integer
- +getHargaItem() : Integer
- +getEmailPemesan() : String
- +getNamaPemesan() : String
- +getTglPemesanan() : Date
- +getAlamatPemesan() : String
- +getTelponPemesan() : Integer

Method-method berikut merupakan *mutator* untuk atribut *class* DataPemesanan :

- -setMerekItem(merek_item : String)
- -getKodeItem(kode_item : Integer)
- -setJenisItem(jenis_item : String)
- -setJumlahItem(jumlah_item : Integer)
- -setHargaItem(harga_item : Integer)
- -setEmailPemesan(email_pemesan : String)
- -setNamaPemesan(nama_pemesan : String)
- -setTglPemesanan(tgl_pemesanan : Date)
- -setAlamatPemesan(alamat_pemesan : String)
- -setTelponPemesan(telpon_pemesan : Integer)

2.2.3.3. Class UIPemesanan



Gambar 2.5 *Class* UIPemesanan

- **Deskripsi**

  Class ini merupakan GUI untuk memesan komputer (UC-Psn_ARK02).

- **Atribut**

  -

- **Method**

  - +displayFormulir()

    menampilkan formulir pemesanan

- `+displayKonfirmasi() : String`

  menampilkan pesan konfirmasi pengecekan kelengkapan formulir

  Return:

  String berisi pesan konfirmasi formulir belum diisi dengan lengkap

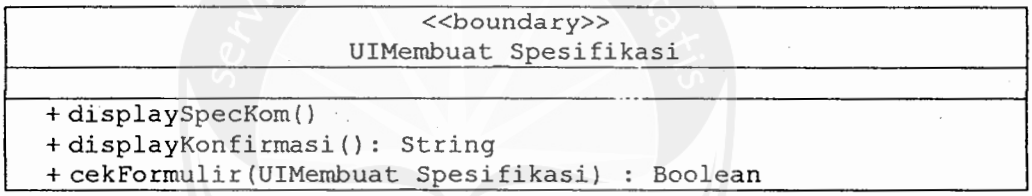
- `+cekFormulir(UIPemesanan) : Boolean`

  Mengecek kelengkapan formulir

  Return:

  True jika informasi penting yang terdapat dalam formulir telah terisi semua

2.2.3.4. Class UIMembuat Spesifikasi Komputer



Gambar 2.6 *Class* UIMembuat_Spesifikasi

- **Deskripsi**

  Class ini merupakan GUI untuk memesan komputer yang spesifikasinya ditentukan oleh pelanggan sendiri.

- **Atribut**

  -

- **Method**

  - `+displaySpecKom()`

    Menampilkan spesifikasi komputer yang dipesan pelanggan

  - `+displayKonfirmasi():String`

    menampilkan pesan konfirmasi pengecekan kelengkapan formulir

    Return:

String berisi pesan konfirmasi formulir belum diisi dengan lengkap

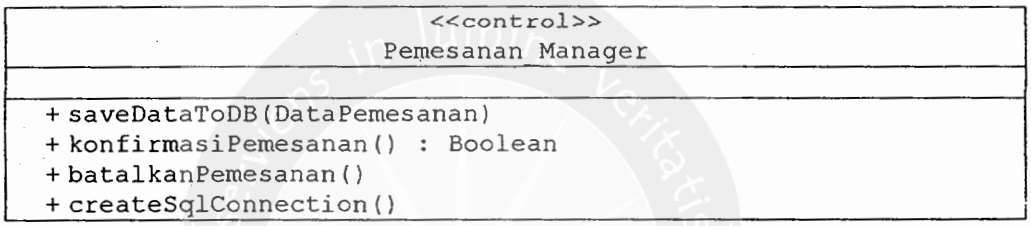
- `+cekFormulir(UIMembuat_Spesifikasi) : Boolean`

  Mengecek kelengkapan formulir

  Return:

  True jika informasi penting yang terdapat dalam formulir telah terisi semua

2.2.3.5. Class Pemesanan Manager



Gambar 2.7 *Class* Pemesanan Manager

- **Deskripsi**

  Class yang berperan sebagai *control class* untuk masalah yang berhubungan dengan pemesanan produk pada sistem ARKAJAX.

- **Atribut**

  -

- **Method**

  - `+saveDataToDB(data_pemesanan)`

    menyimpan data pemesanan ke dalam *database*

  - `+konfirmasiPemesanan() : String`

    Menampilkan pesan konfirmasi pengecekan kelengkapan formulir

    Return:

    String berisi pesan konfirmasi apakah formulir pemesanan sudah lengkap atau belum

  - `+batalkanPemesanan()`

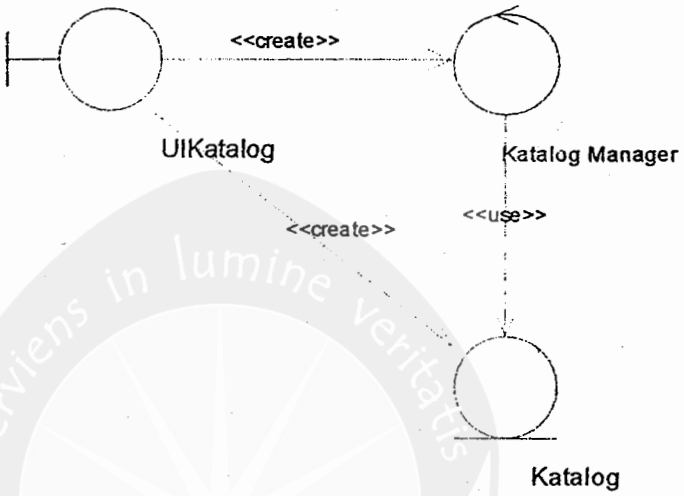
Membatalkan pemesanan produk

- `+createSqlConnection()`

  Membuat koneksi ke *database*

2.2.4. Package Katalog

2.2.4.1. Class Diagram Package Katalog



Gambar 2.8 *Class Diagram Package* Katalog

2.2.4.2. Class UIKatalog

<<boundary>> UIKatalog
+displayKatalog()

Gambar 2.9 *Class* UIKatalog

- **Deskripsi**

  Menampilkan katalog produk yang terdapat dalam sistem ARKAJAX

- **Atribut**

  -

- **Method**

  - `+displayKatalog()`

    menampilkan isi katalog

2.2.4.3. Class Katalog Manager

<<control>> KatalogManager
+cariKatalog(keyword : String)

Gambar 2.10 *Class* KatalogManager

- **Deskripsi**

  Mencari katalog yang diminta oleh *user*

- **Atribut**

  -

- **Method**

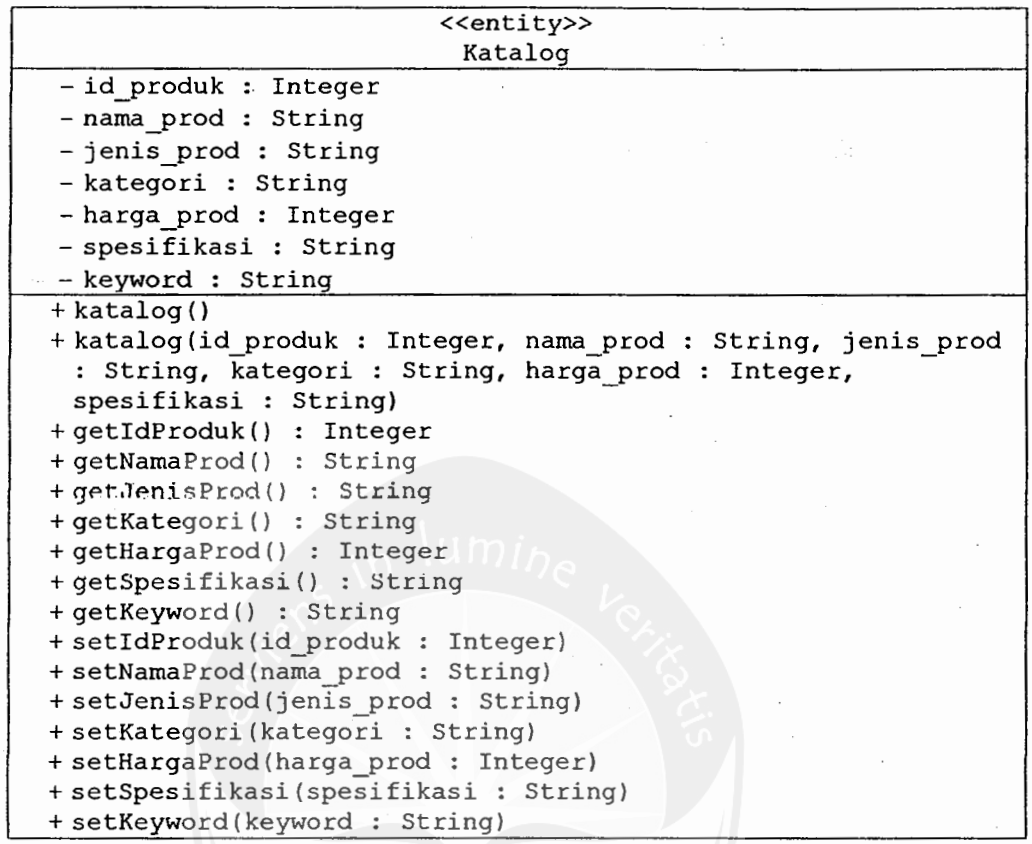
  - `+cariKatalog(keyword : String)`

    Mencari katalog tentang produk-produk yang terdapat dalam sistem ARKAJAX

    Parameter :

    Keyword – kata kunci yang digunakan untuk mencari katalog

2.2.4.4. Class Katalog



Gambar 2.11 *Class* Katalog

- **Deskripsi**

  Class yang mempresentasikan katalog produk yang terdapat dalam sistem ARKAJAX

- **Atribut**
  - `-id_produk : Integer`

    Merepresentasikan kode produk
  - `-nama_prod : String`

    Merepresentasikan merek produk
  - `-jenis_prod : String`

    Merepresentasikan jenis produk
  - `-kategori : String`

    Mereprentasikan kategori produk
  - `-harga_prod : Integer`

Merepresentasikan harga produk

- `-spesifikasi : String`

  Merepresentasikan spesifikasi produk

- **Method**
  - `katalog()`

    Buat *instance* baru `Katalog` tanpa atribut

  - `katalog(id_produk : Integer, nama_prod : String, jenis_prod : String, kategori : String, harga_prod : Integer, spesifikasi : String)`

    Buat *instance* baru `Katalog` dengan atribut terdefinisi

    Parameters :

    `id_produk` – kode produk

    `nama_prod` – merek produk

    `jenis_prod` – jenis produk

    `kategori` – kategori produk

    `harga_prod` – harga produk

    `spesifikasi` – spesifikasi produk

    Method-method berikut merupakan ***accessor*** untuk atribut *class* `Katalog`

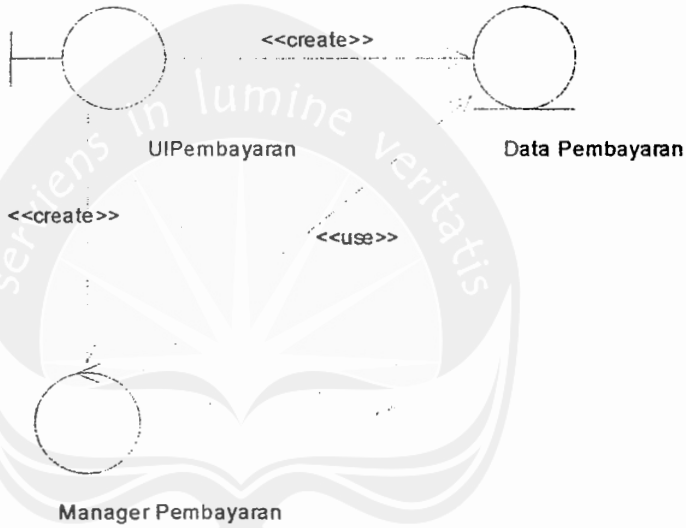
  - `getIdProduk() : Integer`
  - `getNamaProd() : String`
  - `getJenisProd() : String`
  - `getKategori() : String`
  - `getHargaProd() : Integer`
  - `getSpesifikasi() : String`

    Method-method berikut merupakan ***mutator*** untuk atribut *class* `Katalog`

- setIdProduk(id_produk : Integer)
- setNamaProd(nama_prod : String)
- setJenisProd(jenis_prod : String)
- setKategori(kategori : String)
- setHargaProd(harga_prod : Integer)
- setSpesifikasi(spesifikasi : String)

2.2.5. Package Pembayaran

2.2.5.1. Class Diagram Package Pembayaran



Gambar 2.12 *Class Diagram Package* Pembayaran

2.2.5.2. Class UIPembayaran

<<boundary>> UIPembayaran
+ displayKonfirmasi() + cekFormulir(UIPembayaran) : Boolean

Gambar 2.13 *Class* UIPembayaran

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan pembayaran pemesanan komputer (UC-Byr-ARK03).

- **Atribut**

-

- **Method**
  - `+displayKonfirmasi()`

    menampilkan pesan konfirmasi hasil pengecekan kelengkapan formulir

    Return:

    String berisi pesan konfirmasi formulir belum diisi dengan lengkap
  - `+cekFormulir(UIPembayaran) : Boolean`

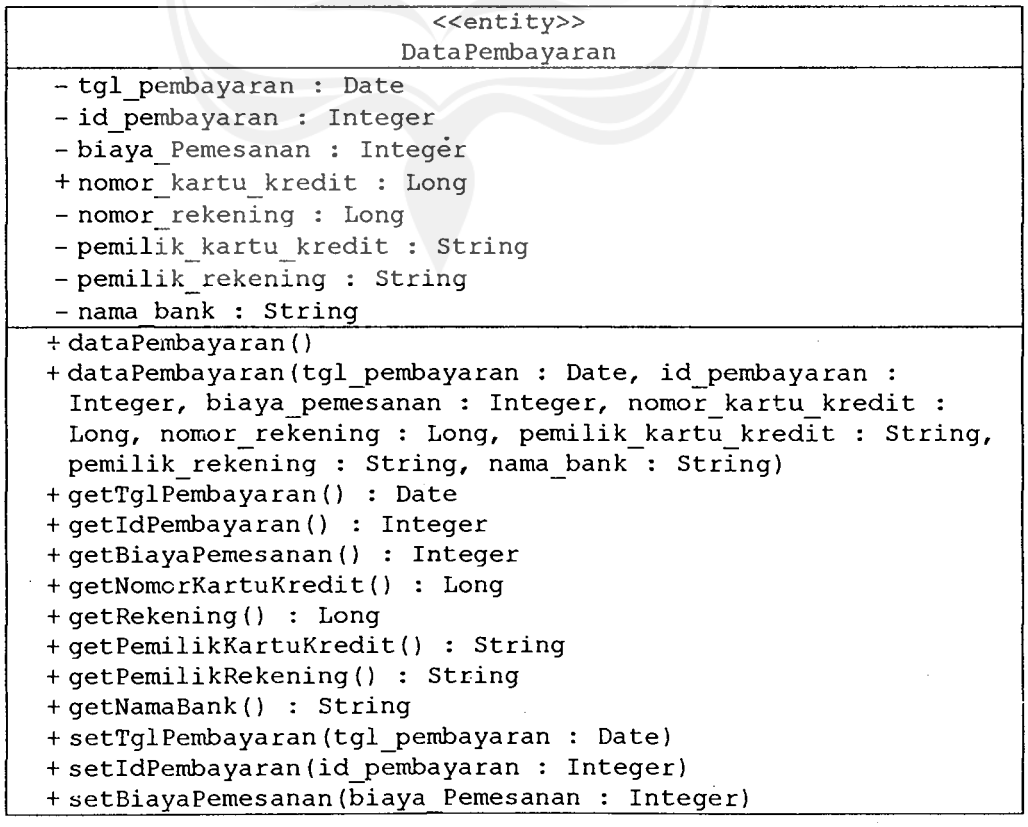
    Boolean

    Mengecek kelengkapan formulir

    Return:

    True jika informasi penting yang terdapat dalam formulir telah terisi semua

2.2.5.3. Class Data Pembayaran



```
+ setNomorKartuKredit(nomor_kartu_kredit : Long)
+ setRekening(nomor_rekening : Long)
+ setPemilikKartuKredit(pemilik_kartu_kredit : String)
+ setPemilikRekening(pemilik_rekening : String)
+ setNamaBank(nama_bank : String)
```

Gambar 2.14 *Class* Data Pembayaran

- **Deskripsi**

  Merepresentasikan transaksi pembayaran atas pemesanan produk pada sistem ARKAJAX

- **Atribut**
  - `-tgl_pembayaran : Date`

    Merepresentasikan tanggal pembayaran
  - `-id_pembayaran : Integer`

    Merepresentasikan kode pembayaran
  - `-biaya_Pemesanan : Integer`

    Merepresentasikan total biaya pemesanan
  - `-nomor_kartu_kredit : Long`

    Merepresentasikan nomor kartu kredit pemesan
  - `-nomor_rekening : Long`

    Merepresentasikan nomor rekening pemesan
  - `-pemilik_kartu_kredit : String`

    Mereprensentasikan nama pemilik kartu kredit
  - `-pemilik_rekening : String`

    Mereprensentasikan nama pemilik kartu kredit
  - `-nama_bank : String`

    Merepresentasikan nama bank penerbit rekening atau kartu kredit
- **Method**
  - `+dataPembayaran()`

    Buat *instance* baru `dataPembayaran` tanpa atribut terdefinisi.

- +dataPembayaran(tgl_pembayaran : Date, id_pembayaran : Integer, biaya_pemesanan : Integer, nomor_kartu_kredit : Long, nomor_rekening : Long, pemilik_kartu_kredit : String, pemilik_rekening : String, nama_bank : String)

  Buat *instance* baru dataPembayaran dengan atribut terdefinisi.

  Parameters :

  tgl_pembayaran – tanggal pembayaran pemesanan

  id_pembayaran – kode pembayaran

  biaya_pemesanan – total biaya pemesanan

  nomor_kartu_kredit – nomor kartu kredit

  nomor_rekening – nomor rekening

  pemilik_kartu_kredit – nama pemilik kartu kredit

  pemilik_rekening – nama pemilik rekening

  nama_bank – nama bank penerbit rekening atau kartu kredit

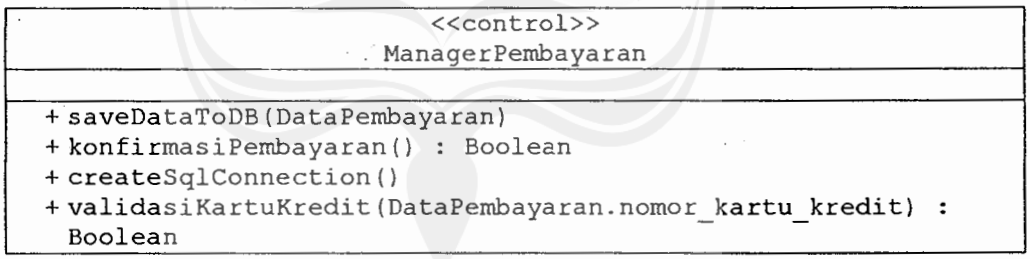
  Method-method berikut merupakan *accessor* untuk atribut *class* DataPembayaran

- +getTglPembayaran() : Date
- +getIdPembayaran() : Integer
- +getBiayaPemesanan() : Integer
- +getNomorKartuKredit() : Long
- +getRekening() : Long
- +getPemilikKartuKredit() : String
- +getPemilikRekening() : String
- +getNamaBank() : String

Method-method berikut merupakan *mutator* untuk atribut *class* DataPembayaran

- +setTglPembayaran(tgl_pembayaran : Date)
- +setIdPembayaran(id_pembayaran : Integer)
- +setBiayaPemesanan(biaya_Pemesanan : Integer)
- +setNomorKartuKredit(nomor_kartu_kredit : Long)
- +setRekening(nomor_rekening : Long)
- +setPemilikKartuKredit(pemilik_kartu_kredit : String)
- +setPemilikRekening(pemilik_rekening : String)
- +setNamaBank(nama_bank : String)

2.2.5.4. Class Manager Pembayaran



Gambar 2.15 *Class* Data Pembayaran

- **Deskripsi**

  Class yang berperan sebagai *control class* untuk masalah yang berhubungan dengan pembayaran pemesanan pada sistem ARKAJAX.

- **Atribut**

  -

- **Method**

- `+validasiKartuKredit(DataPembayaran.nomor_kartu_kredit) : Boolean`

  mengecek validasi kartu kredit

  Return :

  True jika dan hanya jika kartu kredit *valid*

- `+saveDataToDB(DataPembayaran)`

  Menyimpan data pembayaran ke *database*

- `+createSqlConnection()`

  Membuat Koneksi ke SQL server

- `+konfirmasiPembayaran() : String`

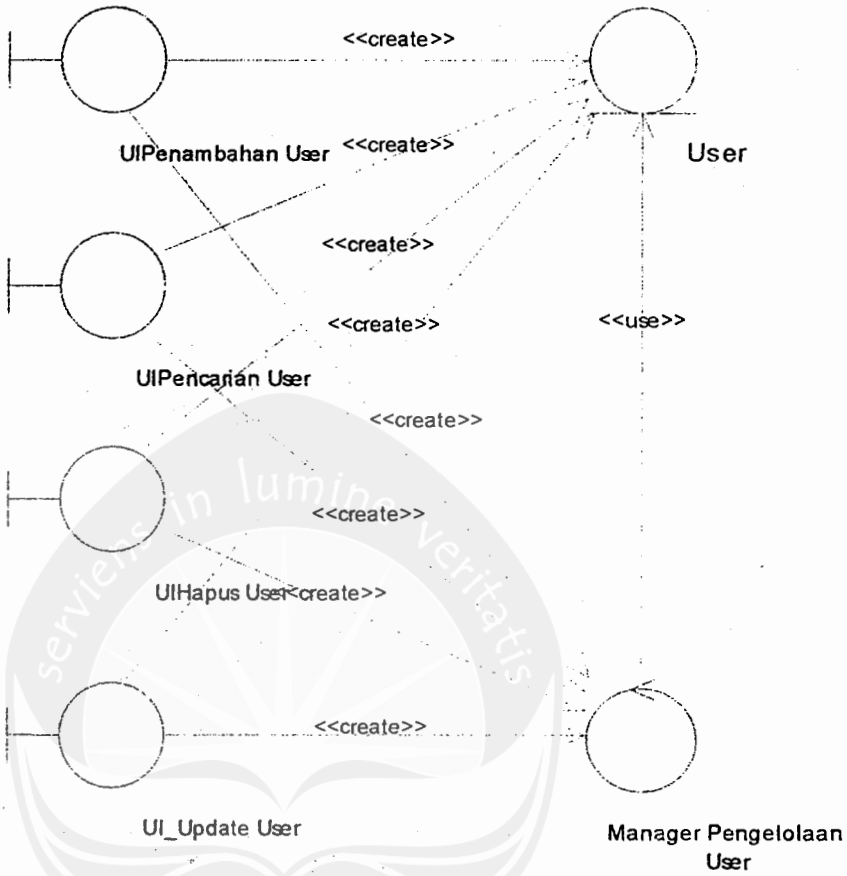
  Menampilkan pesan konfirmasi pengecekan kelengkapan formulir

  Return:

  String berisi pesan konfirmasi apakah formulir pemesanan sudah lengkap atau belum

2.2.6. Package Pengelolaan User

2.2.6.1. *Class Diagram Package* Pengelolaan User



Gambar 2.16. *Class Diagram Package* Pengelolaan User

2.2.6.2. Class UIPenambahan_User

<<boundary>> UIPenambahan_User
+ DisplayKonfirmasiPenambahan():String

Gambar 2.17 Class UIPenambahan_User

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan penambahan user (UC-User-ARK05)

- **Atribut**

  -

- **Method**

+ DisplayKonfirmasiPenambahan():String

menampilkan pesan konfirmasi hasil penambahan user

Return:

String berisi pesan konfirmasi penambahan user

2.2.6.3. Class UIPencarian_User

<<boundary>> UIPencarian_User
+ DisplayUser():Object

Gambar 2.18 Class UIPencarian_User

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan pencarian user (UC-User-ARK05)

- **Atribut**

  -

- **Method**

  + DisplayUser():Object

  Menampilkan data user sebagai hasil pencarian user

  Return:

  Object tabel hasil pencarian data user

2.2.6.4. Class UIHapusUser

<<boundary>> UIHapus_User
+ KonfimasiDeleteUser():String

Gambar 2.19 Class UIHapusUser

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan penghapusan user (UC-User-ARK05)

- **Atribut**

-

- **Method**

  + `KonfimasiDeleteUser():String`

    menampilkan pesan konfirmasi proses menghapus data user

    Return:

    String berisi pesan konfirmasi proses menghapus data user

2.2.6.5. Class UI_Update_User

<<boundary>> UI_Update_User
+ DisplayKonfirmasiUpdate():String

Gambar 2.20 Class UI_Update_User

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan update user (UC-User-ARK05)

- **Atribut**

  -

- **Method**

  + `DisplayKonfirmasiUpdate():String`

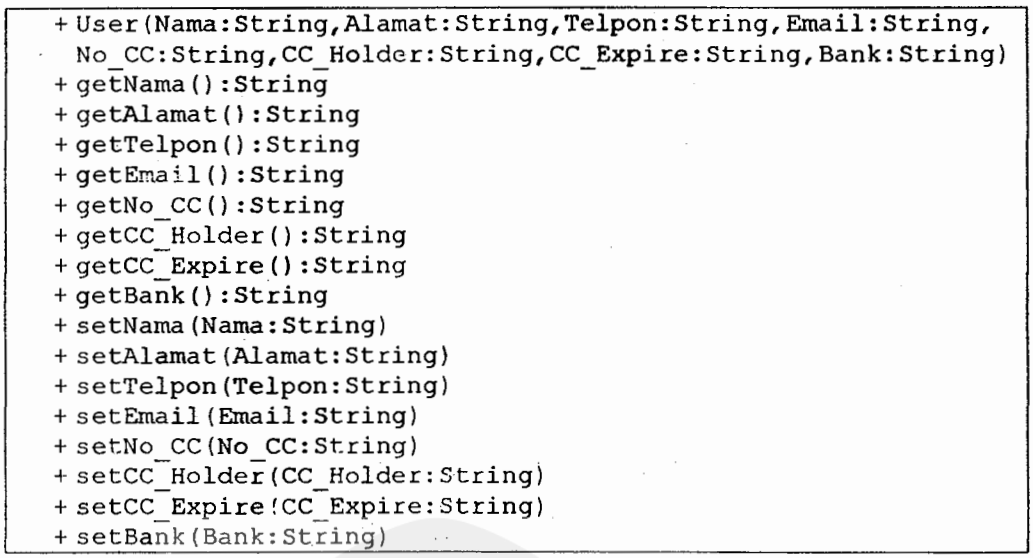
    menampilkan pesan konfirmasi proses update user

    Return:

    String berisi pesan konfirmasi proses update user

2.2.6.6. Class User

<<entity>> User
- Nama:String - Alamat:String - Telpon:String - Email:String - No_CC:String - CC_Holder:String - CC_Expire:String - Bank:String
+ User()



Gambar 2.21 Class User

- **Deskripsi**

  Mempresentasikan user yang telah melakukan transaksi pada sistem ARKAJAX

- **Atribut**
  - `-Nama:String`

    Mempresentasikan nama user
  - `-Alamat:String`

    Mempresentasikan alamat tempat tinggal user
  - `-Telpon:String`

    Mempresentasikan nomor telepon user
  - `-Email:String`

    Mempresentasikan email user
  - `-No_CC:String`

    Mempresentasikan nomor kartu kredit yang digunakan user
  - `-CC_Holder:String`

    Mempresentasikan nama pemegang kartu kredit
  - `-CC_Expire:String`

    Mempresentasikan tanggal kadaluarsa kartu kredit
  - `-Bank:String`

Mempresentasikan nama bank yang digunakan untuk melakukan transaksi

- **Method**
  - `+User()`

    Buat *instance* baru `user` tanpa atribut terdefinisi
  - `+User(Nama:String,Alamat:String,Telpon:String,Email:String,No_CC:String,CC_Holder:String,CC_Expire:String,Bank:String)`

    Buat *instance* baru `user` dengan atribut terdefinisi

    `Parameter`

    `Nama:String` - Mempresentasikan nama user

    `Alamat:String` - Mempresentasikan alamat tempat tinggal user

    `Telpon:String` - Mempresentasikan nomor telepon user

    `Email:String` - Mempresentasikan email user

    `No_CC:String` - Mempresentasikan nomor kartu kredit yang digunakan user

    `CC_Holder:String` - Mempresentasikan nama pemegang kartu kredit

    `CC_Expire:String` - Mempresentasikan tanggal kadaluarsa kartu kredit

    `Bank:String` - Mempresentasikan nama bank yang digunakan untuk melakukan transaksi

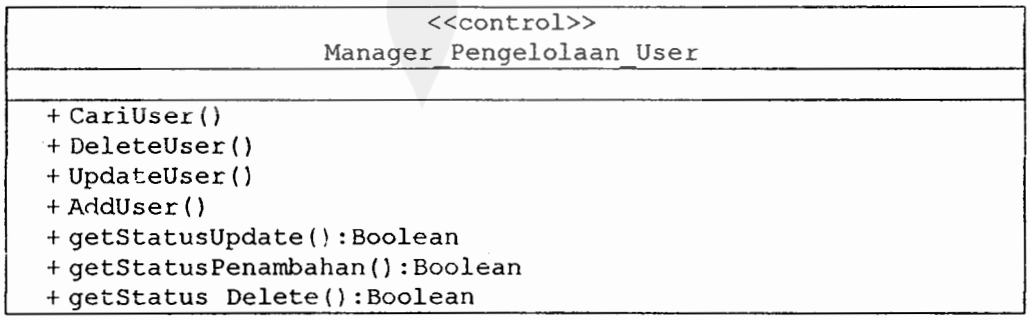
    Method-method berikut merupakan *accessor* untuk atribut *class* `User`
  - `+getNama():String`

- +getAlamat():String
- +getTelpon():String
- +getEmail():String
- +getNo_CC():String
- +getCC_Holder():String
- +getCC_Expire():String
- +getBank():String

Method-method berikut merupakan *mutator* untuk atribut *class* User

- +setNama(Nama:String)
- +setAlamat(Alamat:String)
- +setEmail(Email:String)
- +setTelpon(Telpon:String)
- +setNo_CC(No_CC:String)
- +setCC_Holder(CC_Holder:String)
- +setCC_Expire(Expire:String)

2.2.6.7. Class Manager_Pengelolaan_User



Gambar 2.22 Class Manager_Pengelolaan_User

- **Deskripsi**

  Class yang berperan sebagai *control class* untuk masalah yang berhubungan dengan pengelolaan user pada sistem ARKAJAX

- **Atribut**

  -

- **Method**

  - `+CariUser()`

    Mencari data user

  - `+DeleteUser()`

    Menghapus data user

  - `+UpdateUser()`

    Meng-update data user

  - `+AddUser()`

    Menambahkan data user ke dalam database

  - `+getStatusUpdate():Boolean`

    Mengecek apakah proses update user sukses atau gagal

    `Return`

    Jika *true* maka sistem akan menampilkan pesan bahwa data telah sukses di-update, jika false maka sistem menampilkan pesan bahwa data gagal di-update

  - `+getStatusPenambahan():Boolean`

    Mengecek apakah proses penambahan user sukses atau gagal

    `Return`

    Jika *true* maka sistem akan menampilkan pesan bahwa proses penambahan data user sukses, jika false maka sistem menampilkan pesan bahwa proses penambahan data user gagal

  - `+getStatusDelete():Boolean`

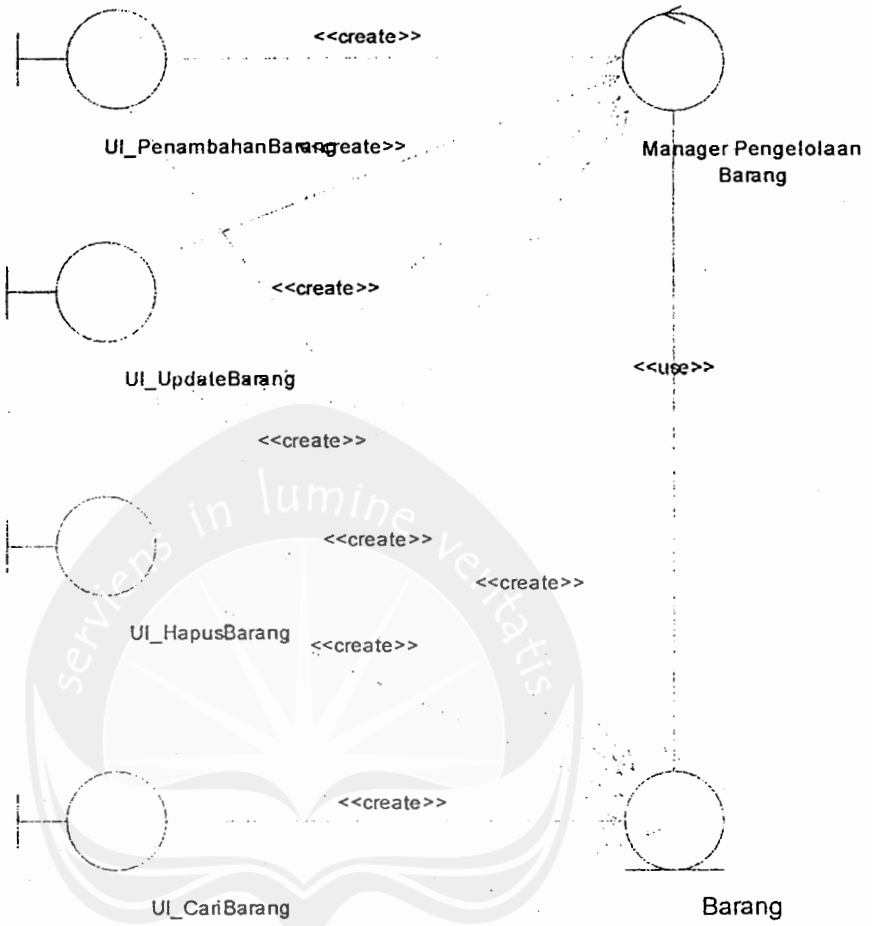
    Mengecek apakah proses delete user sukses atau gagal

    `Return`

    Jika *true* maka sistem akan menampilkan pesan bahwa data telah sukses di-delete, jika false maka sistem menampilkan pesan bahwa data gagal di-delete

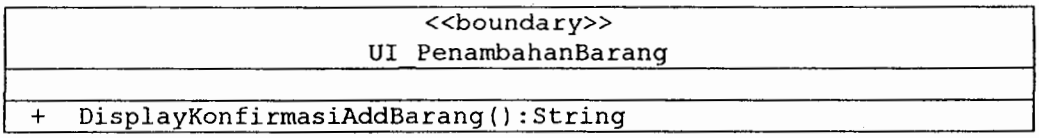
2.2.7. Package Pengelolaan Barang

2.2.7.1. *Class Diagram Package* Pengelolaan Barang



Gambar 2.23 *Class Diagram Package* Pengelolaan Barang

2.2.7.2. Class UI_PenambahanBarang



Gambar 2.24 Class UI_PenambahanBarang

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan penambahan barang (UC-User-ARK06)

- **Atribut**

  -

- **Method**

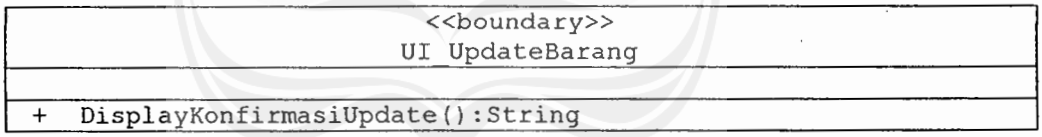
  + `DisplayKonfirmasiAddBarang():String`

  menampilkan pesan konfirmasi proses penambahan barang

  Return:

  String berisi pesan konfirmasi proses penambahan barang

2.2.7.3. Class UI_UpdateBarang



Gambar 2.25 Class UI_UpdateBarang

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan update barang (UC-User-ARK06)

- **Atribut**

  -

- **Method**

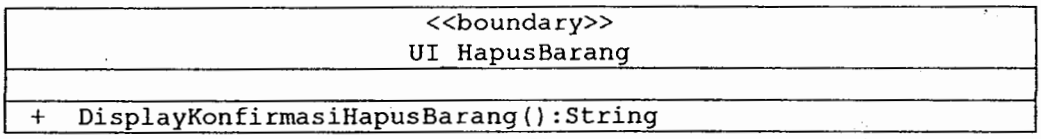
  + `DisplayKonfirmasiUpdate():String`

  menampilkan pesan konfirmasi proses update barang

  Return:

  String berisi pesan konfirmasi proses update barang

2.2.7.4. Class UI_HapusBarang



Gambar 2.26 Class UI_HapusBarang

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan delete barang (UC-User-ARK06)

- **Atribut**

  -

- **Method**

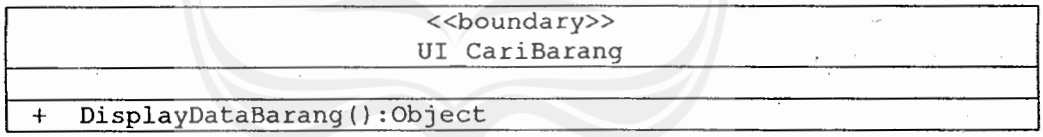
  + `DisplayKonfirmasiHapusBarang():String`

  menampilkan pesan konfirmasi proses delete barang

  Return:

  String berisi pesan konfirmasi proses delete barang

2.2.7.5. Class UI_CariBarang



Gambar 2.27 Class UI_CariBarang

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan pencarian barang (UC-User-ARK06)

- **Atribut**

  -

- **Method**

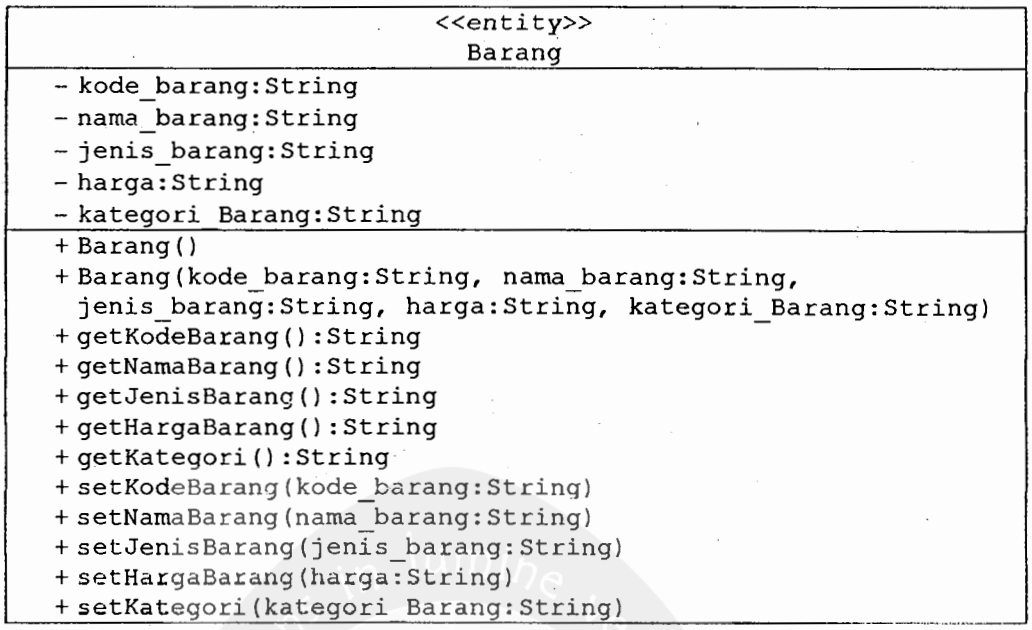
  + `DisplayDataBarang():Object`

  Menampilkan data barang sebagai hasil pencarian barang

  Return:

  Object adalah tabel data barang

2.2.7.6. Class Barang



Gambar 2.28 Class Barang

- **Deskripsi**

  Mempresentasikan barang yang dijual pada sistem ARKAJAX

- **Atribut**
  - `-kode_barang:String`

    Mempresentasikan kode barang
  - `-nama_barang:String`

    Mempresentasikan nama barang
  - `-jenis_barang:String`

    Mempresentasikan jenis barang
  - `-harga:String`

    Mempresentasikan harga barang
  - `-kategori_Barang:String`

    Mempresentasikan kategori barang

- **Method**
  - `+Barang()`

    Buat *instance* baru `barang` tanpa atribut terdefinisi
  - `+Barang(kode_barang:String, nama_barang:String, jenis_barang:String, harga:String, kategori_Barang:String)`

    Buat *instance* baru `barang` dengan atribut terdefinisi

    `Parameter`

    `kode_barang:String` - Mempresentasikan kode barang

    `nama_barang:String` - Mempresentasikan nama barang

    `jenis_barang:String` - Mempresentasikan jenis barang

    `harga:String` - Mempresentasikan harga barang

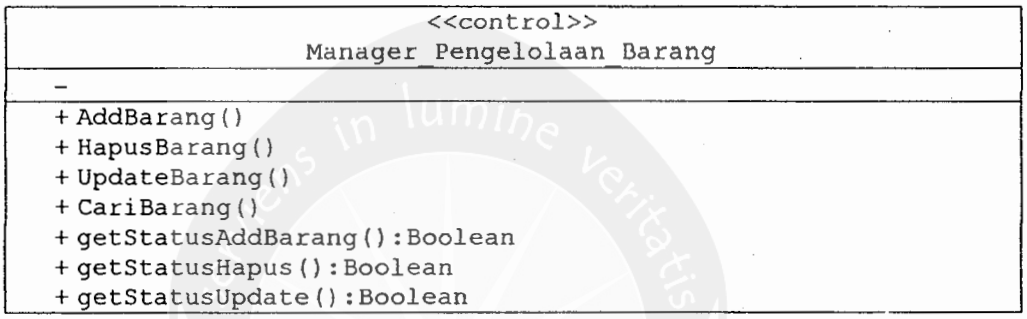
    `kategori_Barang:String` - Mempresentasikan kategori barang

    Method-method berikut merupakan *accessor* untuk atribut *class* `Barang`
  - `+getKodeBarang():String`
  - `+getNamaBarang():String`
  - `+getJenisBarang():String`
  - `+getHargaBarang():String`
  - `+getKategori():String`

Method-method berikut merupakan *mutator* untuk atribut *class* Barang

- +setKodeBarang(kode_barang:String)
- +setNamaBarang(nama_barang:String)
- +setJenisBarang(jenis_barang:String)
- +setHargaBarang(harga:String)
- +setKategori(kategori_Barang:String)

2.2.7.7. Class Manager_Pengelolaan_Barang



Gambar 2.29 Class Manager_Pengelolaan_Barang

- **Deskripsi**

  Class yang berperan sebagai *control class* untuk masalah yang berhubungan dengan pengelolaan user pada sistem ARKAJAX

- **Atribut**

  -

- **Method**

  - +AddBarang()

    Menambahkan data barang ke dalam database

  - +HapusBarang()

    Menghapus data barang

  - +UpdateBarang()

    Meng-update data barang

  - +CariBarang()

Mencari data barang

- `+getStatusAddBarang():Boolean`

  Mengecek apakah proses add barang sukses atau gagal

  `Return`

  Jika *true* maka sistem akan menampilkan pesan bahwa data telah sukses ditambahkan, jika false maka sistem menampilkan pesan bahwa data gagal ditambahkan

- `+getStatusHapus():Boolean`

  Mengecek apakah proses delete barang sukses atau gagal

  `Return`

  Jika *true* maka sistem akan menampilkan pesan bahwa data telah sukses di-delete, jika false maka sistem menampilkan pesan bahwa data gagal di-delete

- `+getStatusUpdate():Boolean`

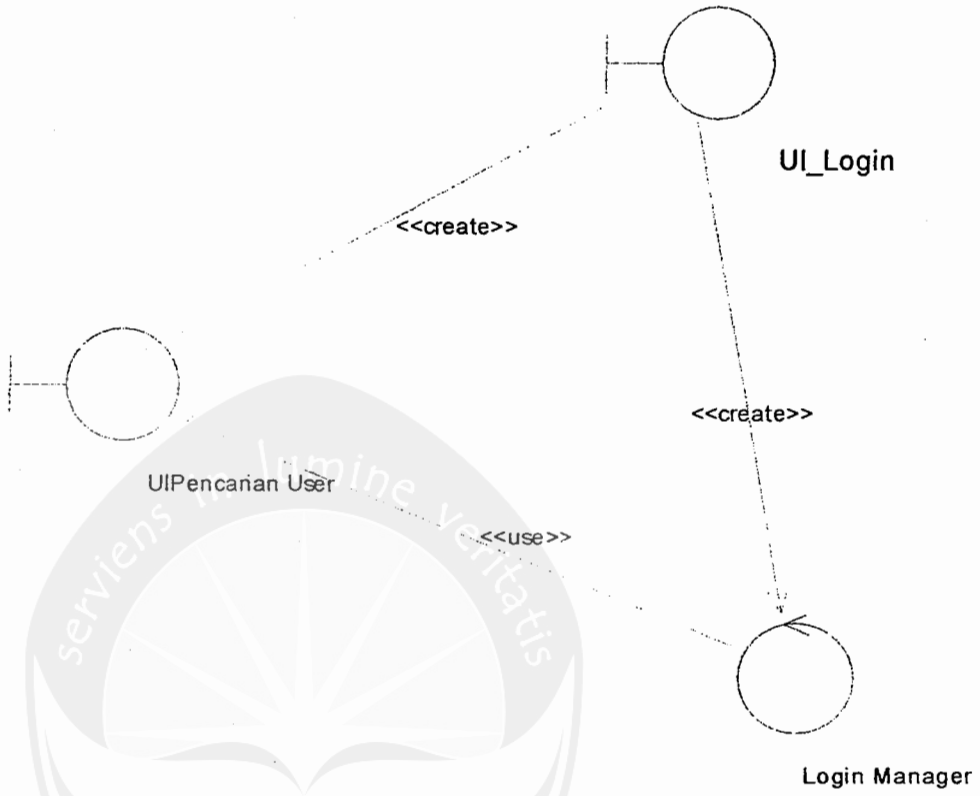
  Mengecek apakah proses update barang sukses atau gagal

  `Return`

  Jika *true* maka sistem akan menampilkan pesan bahwa data telah sukses di-update, jika false maka sistem menampilkan pesan bahwa data gagal di-update

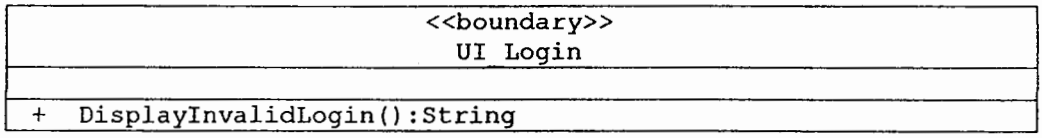
2.2.8. Package Login

2.2.8.1. *Class Diagram Package* Login



Gambar 2.30 *Class Diagram Package Login*

2.2.8.2. Class UI_Login



Gambar 2.31 Class UI_Login

- **Deskripsi**

  Class ini merupakan GUI untuk melakukan login (UC-User-ARK07)

- **Atribut**

  -

- **Method**

  + `DisplayInvalidLogin():String`

  Menampilkan pesan konfirmasi proses login

  Return:

  String adalah pesan konfirmasi jika user salah memasukkan username atau password

2.2.8.3. Class Login_Manager

<<control>> Manager_Pengelolaan_Barang
-
+ ValidasiUsernamePassword():Boolean

Gambar 2.32 Class Manager_Pengelolaan_Barang

- **Deskripsi**

  Class yang berperan sebagai *control class* untuk masalah yang berhubungan dengan login administrator pada sistem ARKAJAX

- **Atribut**

  -

- **Method**

  - `+ValidasiUsernamePassword():Boolean`

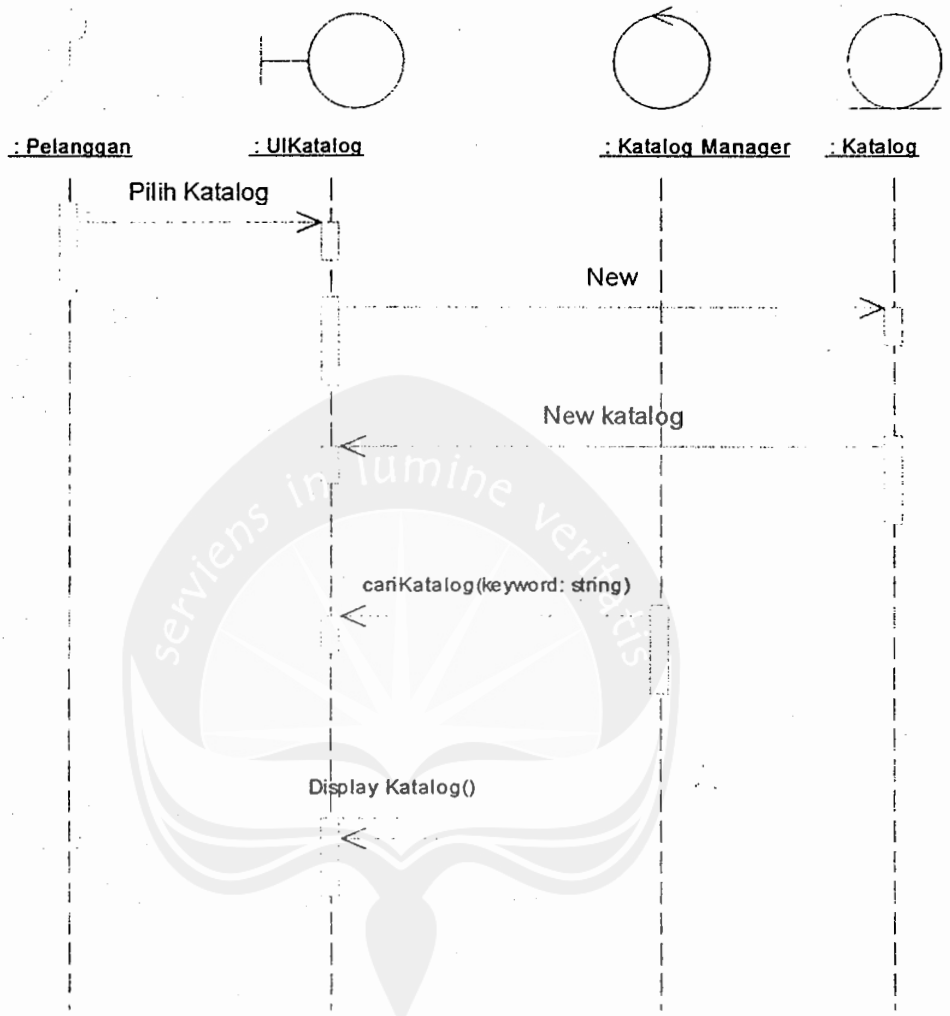
    Mem-validasi username dan password

    `Return`

    Jika *true* maka username dan password valid, jika false maka username dan password invalid

2.3. Realisasi Use Case

2.3.1. Use Case : Browsing Katalog

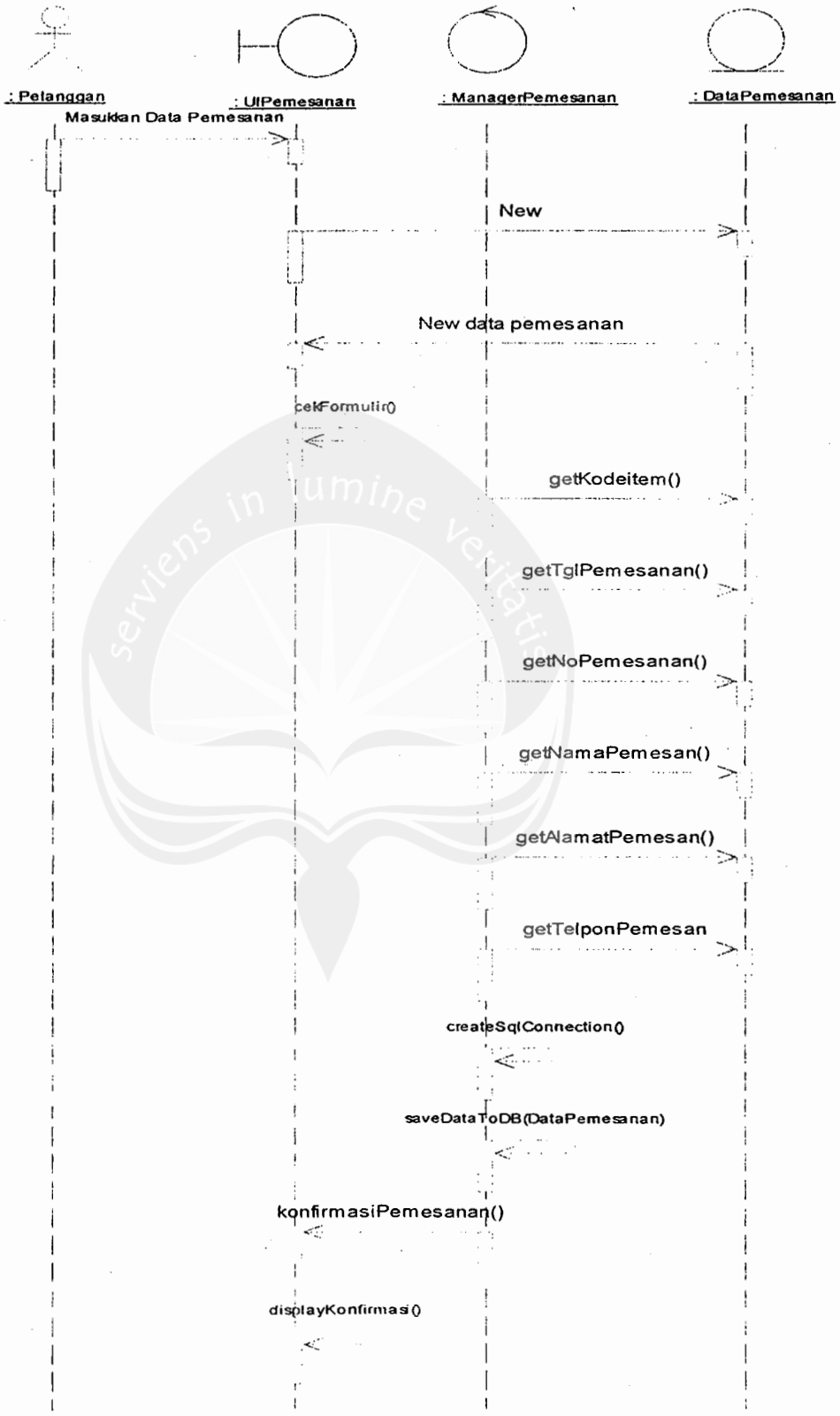


Gambar 2.33 *Use Case Browsing* Katalog

Flow of Events :

1. Pelanggan memilih katalog yang akan dilihat melalui *boundary class* `UIKatalog`
2. Pelanggan memicu terjadinya pembentukan objek `katalog` yang merupakan *instance* dari `class katalog`
3. Objek katalog terbentuk
4. Pelanggan melakukan pencarian katalog
5. Sistem menampilkan katalog yang dicari oleh pelanggan

2.3.2. Use Case : Memesan Komputer



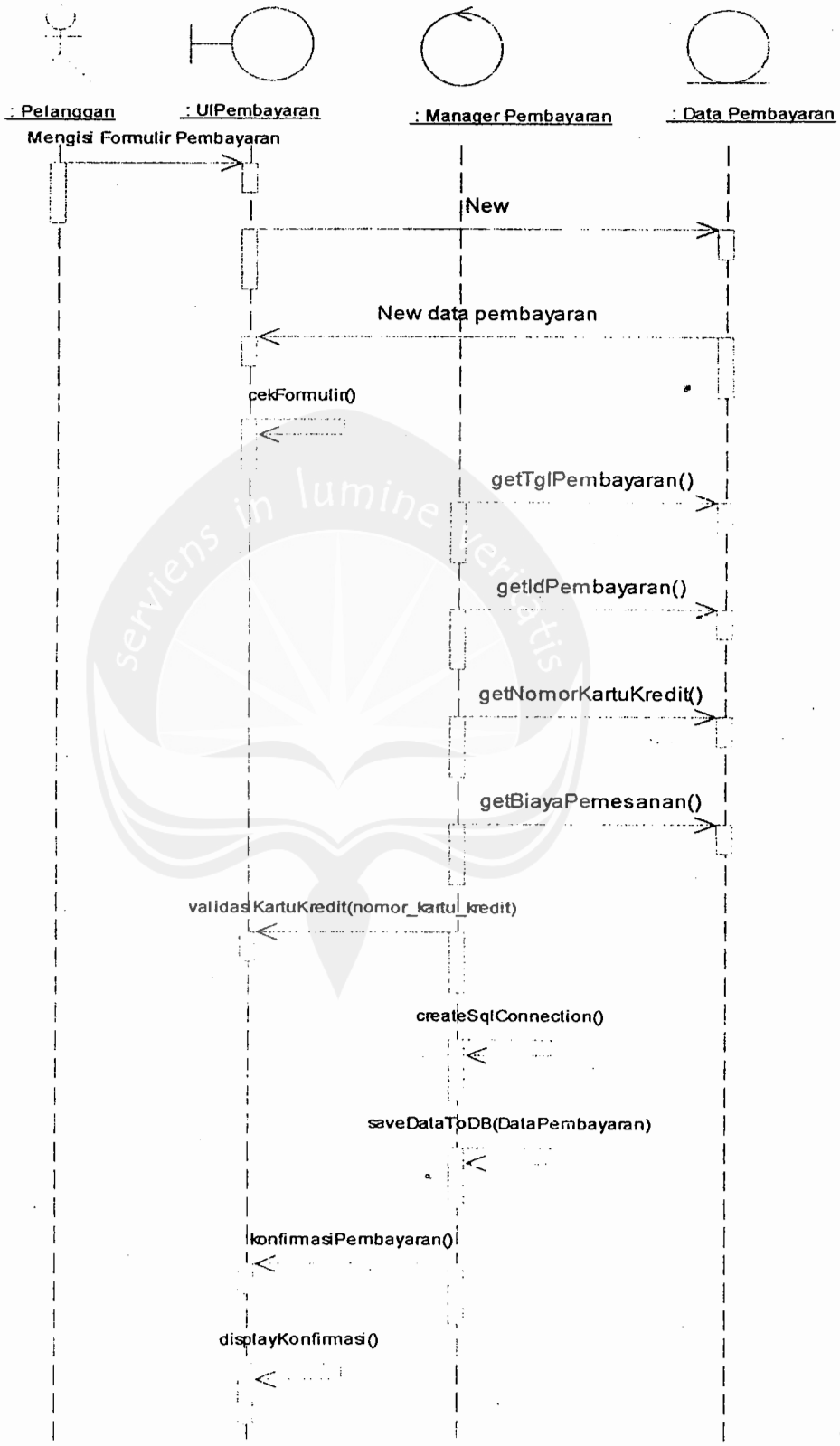
Gambar 2.34 *Use Case* Memesan Komputer

Flow of Events :

1. Pelanggan memilih katalog yang akan dilihat melalui *boundary class* `UIPemesanan`
2. Pelanggan memicu terjadinya pembentukan objek `data pemesanan` yang merupakan *instance* dari `class Data Pemesanan`
3. Objek data pemesanan terbentuk
4. Class `UIPemesanan` mengecek kelengkapan formulir menggunakan method `cekFormulir()`
5. Class `ManagerPemesanan` Memanggil method `getKodeitem()` yang terdapat di class `DataPemesanan` untuk mengambil kode item
6. Class `ManagerPemesanan` Memanggil method `getTglPemesanan()` yang terdapat di class `DataPemesanan` untuk mengambil tanggal pemesanan produk
7. Class `ManagerPemesanan` Memanggil method `getNoPemesanan()` yang terdapat di class `DataPemesanan` untuk mengambil data nomor pemesanan
8. Class `ManagerPemesanan` Memanggil method `getNamaPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil nama pemesan produk
9. Class `ManagerPemesanan` Memanggil method `getAlamatPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil alamat pemesan produk
10. Class `ManagerPemesanan` Memanggil method `getTelponPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil telpon pemesan produk
11. Class `ManagerPemesanan` membuka koneksi dengan database server

12. Class `ManagerPemesanan` menyimpan data pemesanan ke dalam *database*, penyimpanannya dilakukan dengan menggunakan method `saveDataToDB(kode_item, tgl_pesan, no_pesan, nama, alamat, telpon)`
13. Class boundary `UIPemesanan` memanggil method `konfirmasiPemesanan()` dari class `ManagerPemesanan` yang memberikan konfirmasi bahwa pemesanan telah selesai selesai
14. Class boundary `UIPemesanan` memanggil method `displayKonfirmasi()` menampilkan konfirmasi pemesanan kepada pelanggan

2.3.3. Use Case : Membayar Pemesanan Komputer



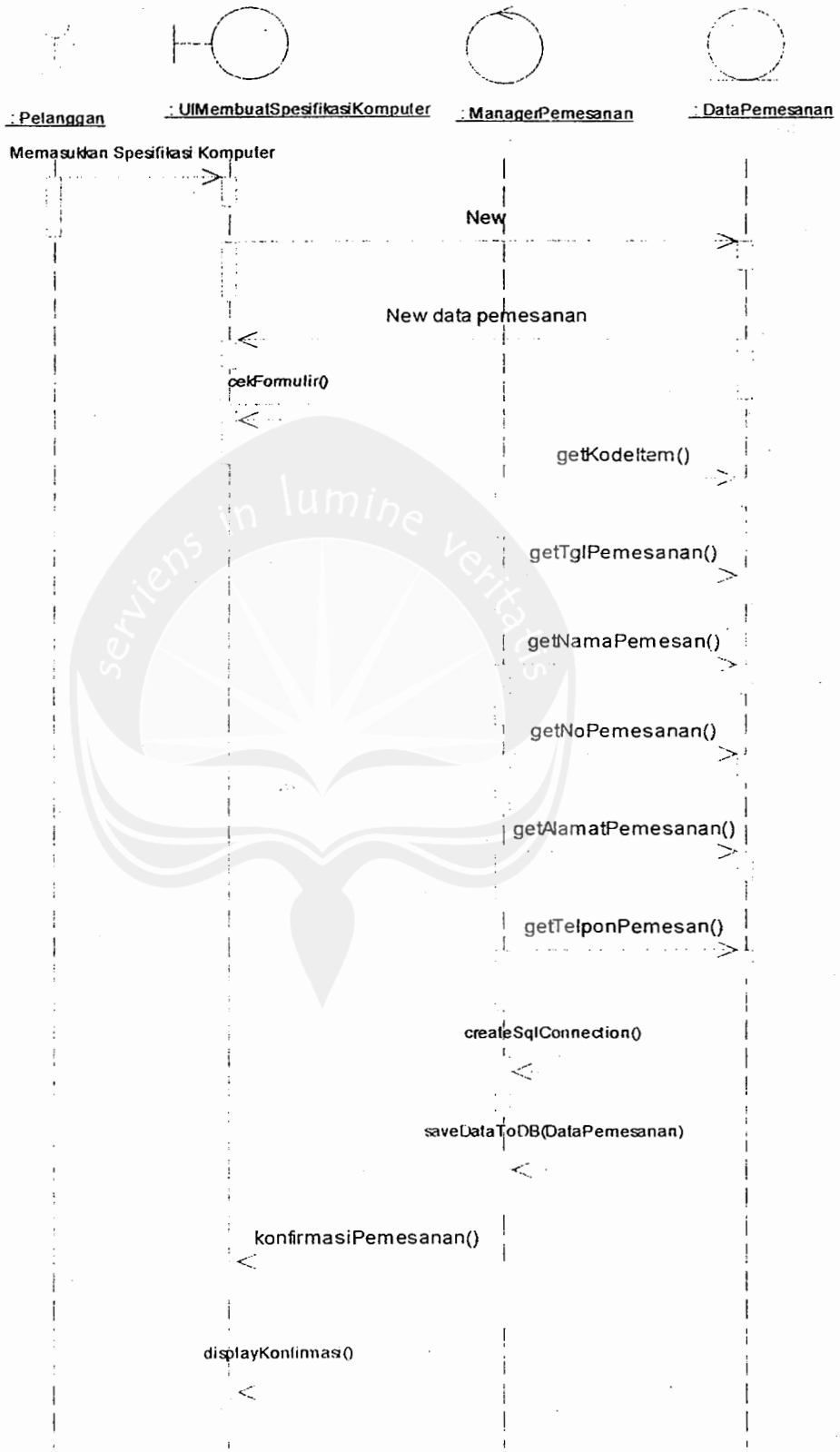
Gambar 2.35 Use Case : Membayar Pemesanan Komputer

Flow of events :

1. Pelanggan memilih katalog yang akan dilihat melalui *boundary class* `UIPembayaran`
2. Pelanggan memicu terjadinya pembentukan objek `data pembayaran` yang merupakan *instance* dari `class Data Pembayaran`
3. Objek data pembayaran terbentuk
4. Class `UIPemesanan` mengecek kelengkapan formulir menggunakan method `cekFormulir()`
5. Class `ManagerPembayaran` Memanggil method `getTglPembayaran()` yang terdapat di class `DataPembayaran` untuk mengambil data tanggal pembayaran pesanan
6. Class `ManagerPembayaran` Memanggil method `getIdPembayaran()` yang terdapat di class `DataPembayaran` untuk mengambil nomor identitas pembayaran
7. Class `ManagerPembayaran` Memanggil method `getNomorKartuKredit()` yang terdapat di class `DataPembayaran` untuk mengambil nomor kartu kredit pelanggan
8. Class `ManagerPembayaran` Memanggil method `getBiayaPemesanan()` yang terdapat di class `DataPembayaran` untuk mengambil informasi total biaya belanja
9. Validasi kartu kredit dilakukan oleh class `UIPembayaran` dengan cara memanggil method `validasiKartuKredit(nomor_kartu_kredit)` yang terdapat didalam class `ManagerPembayaran`

10. Class `ManagerPemesanan` membuka koneksi dengan database server, dengan menggunakan method `createSqlConnection()`
11. Class `ManagerPembayaran` menyimpan data pemesanan ke dalam *database*, penyimpanannya dilakukan dengan menggunakan method `saveDataToDB(Tgl_bayar, id_bayar, nomor_ccard, biaya_pesan)`
12. Class boundary `UIPembayaran` memanggil method `konfirmasiPemesanan()` dari class `ManagerPembayaran` yang memberikan konfirmasi bahwa pemesanan telah selesai selesai
13. Class boundary `UIPembayaran` memanggil method `displayKonfirmasi()` menampilkan konfirmasi pemesanan kepada pelanggan

2.3.4. Use Case : Membuat Spesifikasi Komputer



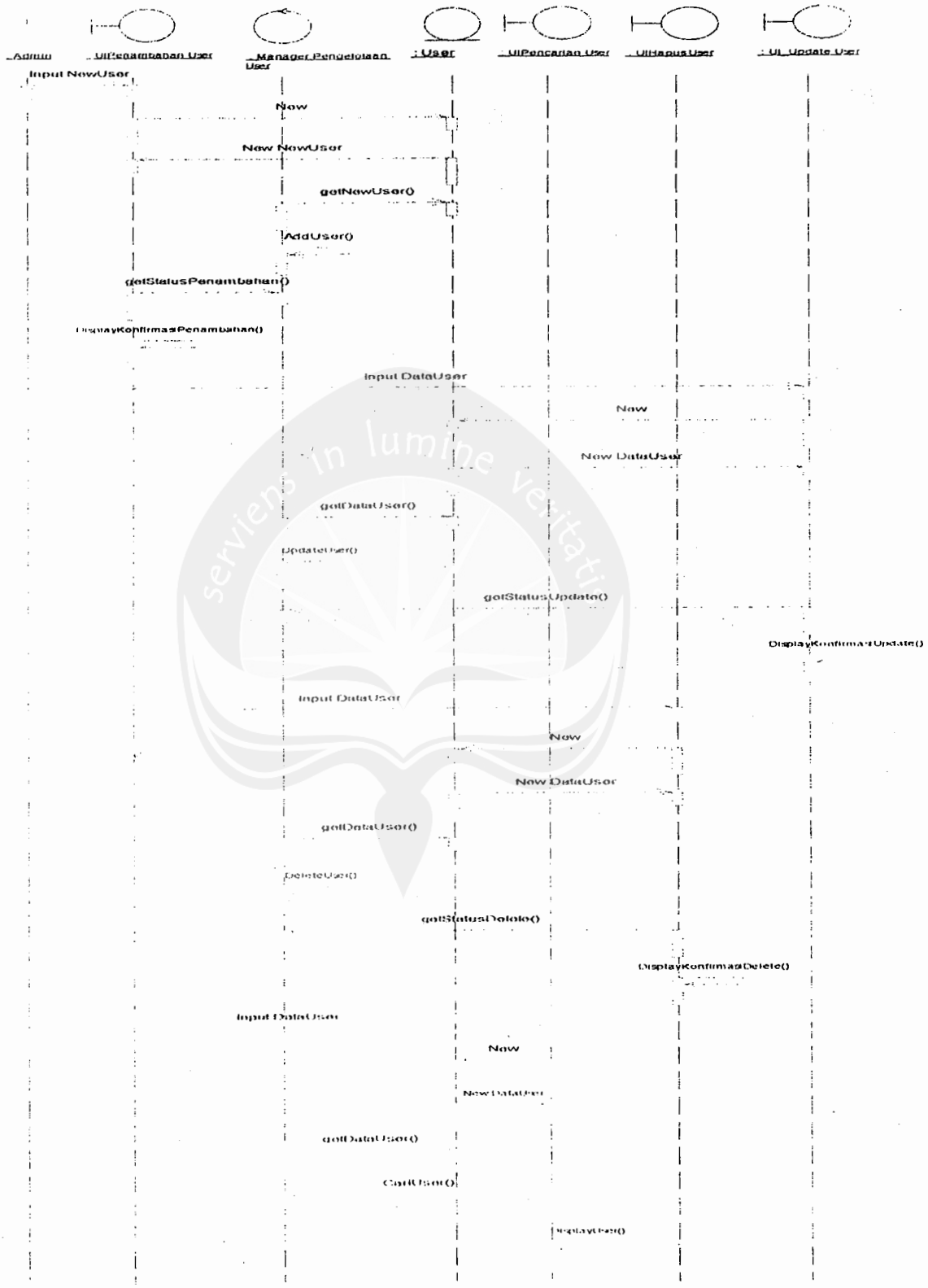
Gambar 2.36 Use Case : Membuat Spesifikasi Komputer

Flow of events :

1. Pelanggan memilih katalog yang akan dilihat melalui *boundary class* `UIMembuatSpesifikasiKomputer`
2. Pelanggan memicu terjadinya pembentukan objek `data pemesanan` yang merupakan *instance* dari `class Data Pemesanan`
3. Objek data pemesanan terbentuk
4. Class `UIMembuatSpesifikasiKomputer` mengecek kelengkapan formulir menggunakan method `cekFormulir()`
5. Class `ManagerPemesanan` Memanggil method `getKodeitem()` yang terdapat di class `DataPemesanan` untuk mengambil kode item
6. Class `ManagerPemesanan` Memanggil method `getTglPemesanan()` yang terdapat di class `DataPemesanan` untuk mengambil tanggal pemesanan
7. Class `ManagerPemesanan` Memanggil method `getNamaPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil nama pemesan item
8. Class `ManagerPemesanan` Memanggil method `getNoPemesanan()` yang terdapat di class `DataPemesanan` untuk mengambil data nomor pemesanan
9. Class `ManagerPemesanan` Memanggil method `getAlamatPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil alamat pemesan produk
10. Class `ManagerPemesanan` Memanggil method `getTelponPemesan()` yang terdapat di class `DataPemesanan` untuk mengambil telpon pemesan produk

11. Class `ManagerPemesanan` membuka koneksi dengan database server, dengan menggunakan method `createSqlConnection()`
12. Class `ManagerPemesanan` menyimpan data pemesanan ke dalam *database*, penyimpanannya dilakukan dengan menggunakan method `saveDataToDB(kode_item, tgl_pesan, no_pesan, nama, alamat, telpon)`
13. Class boundary `UIMembuatSpesifikasiKomputer` memanggil method `konfirmasiPemesanan()` dari class `ManagerPemesanan` yang memberikan konfirmasi bahwa pemesanan telah selesai selesai
14. Class boundary `UIMembuatSpesifikasiKomputer` memanggil method `displayKonfirmasi()` menampilkan konfirmasi pemesanan kepada pelanggan

2.3.5. Use Case : Pengelolaan User



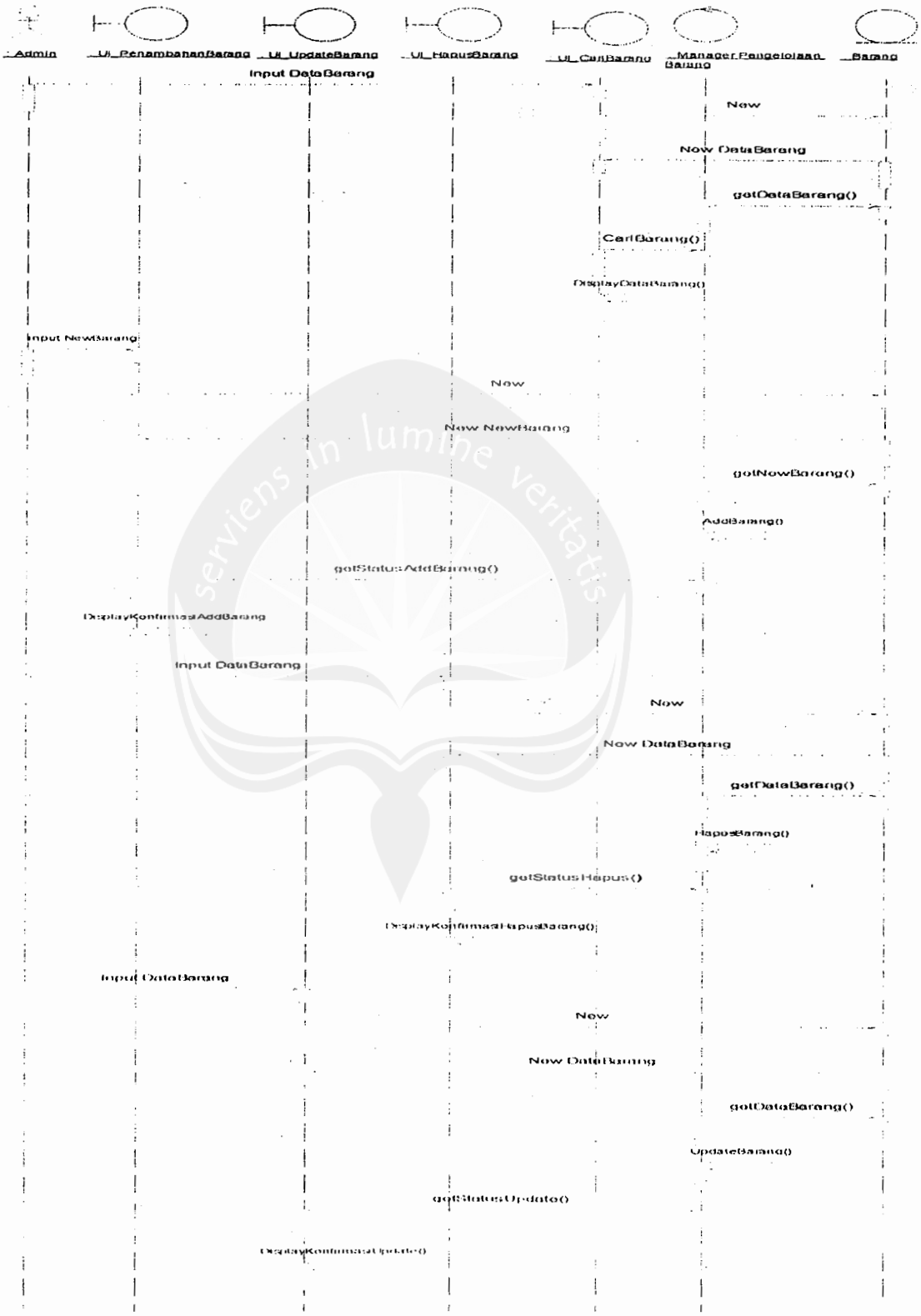
2.37 Use Case : Pengelolaan User

Flow of events:

1. Administrator memasukkan data user yang baru melalui `class boundary UIPenambahanUser`
2. Admin memicu terjadinya pembentukan objek `User` yang merupakan *instance* dari `class User`
3. Objek `User` terbentuk
4. Class `ManagerPengelolaanUser` Memanggil method `getUser()` yang terdapat di class `User` untuk mengambil data user
5. `Class ManagerPengelolaanUser` Menambahkan data user ke dalam database
6. `Class boundary UIPenambahanUser` memanggil `method getStatusPenambahan` yang terdapat didalam `class ManagerPengelolaanUser`, untuk mengetahui apakah proses penambahan berhasil atau gagal
7. `Class boundary UIPenambahanUser` menampilkan pesan konfirmasi penambahan user
8. Administrator memasukkan data user yang baru melalui `class boundary UIUpdateUser`
9. Admin memicu terjadinya pembentukan objek `User` yang merupakan *instance* dari `class User`
10. Objek `User` terbentuk
11. Class `ManagerPengelolaanUser` Memanggil method `getUser()` yang terdapat di class `User` untuk mengambil data user
12. `Class ManagerPengelolaanUser` Mengupdate data user
13. `Class UI_UpdateUser` memanggil method `getStatusUpdate()` untuk mengetahui apakah proses update data user sukses atau gagal

14. `Class boundary UI_UpdateUser` menampilkan pesan konfirmasi update user
15. Administrator memasukkan data user yang baru melalui `class boundary UIHapusUser`
16. Admin memicu terjadinya pembentukan objek `User` yang merupakan *instance* dari `class User`
17. Objek `User` terbentuk
18. Class `ManagerPengelolaanUser` Memanggil method `getUser()` yang terdapat di class `User` untuk mengambil data user
19. `Class ManagerPengelolaanUser` Menghapus data user
20. `Class UIHapusUser` memanggil method `getStatusDelete()` untuk mengetahui apakah proses delete data user sukses atau gagal
21. `Class boundary UIHapusUser` menampilkan pesan konfirmasi delete user
22. Administrator memasukkan data user yang baru melalui `class boundary UIPencarianUser`
23. Admin memicu terjadinya pembentukan objek `User` yang merupakan *instance* dari `class User`
24. Objek `User` terbentuk
25. `Class ManagerPengelolaanUser` Memanggil `method getUser()` yang terdapat di `class User` untuk mengambil data user
26. `Class UIPencarianUser` memanggil `method CariUser()` yang terdapat didalam `class ManagerPengelolaanUser` untuk melakukan pencarian user
27. `Class UIPencarianUser` Menampilkan data user

2.3.6. Use Case : Pengelolaan Barang



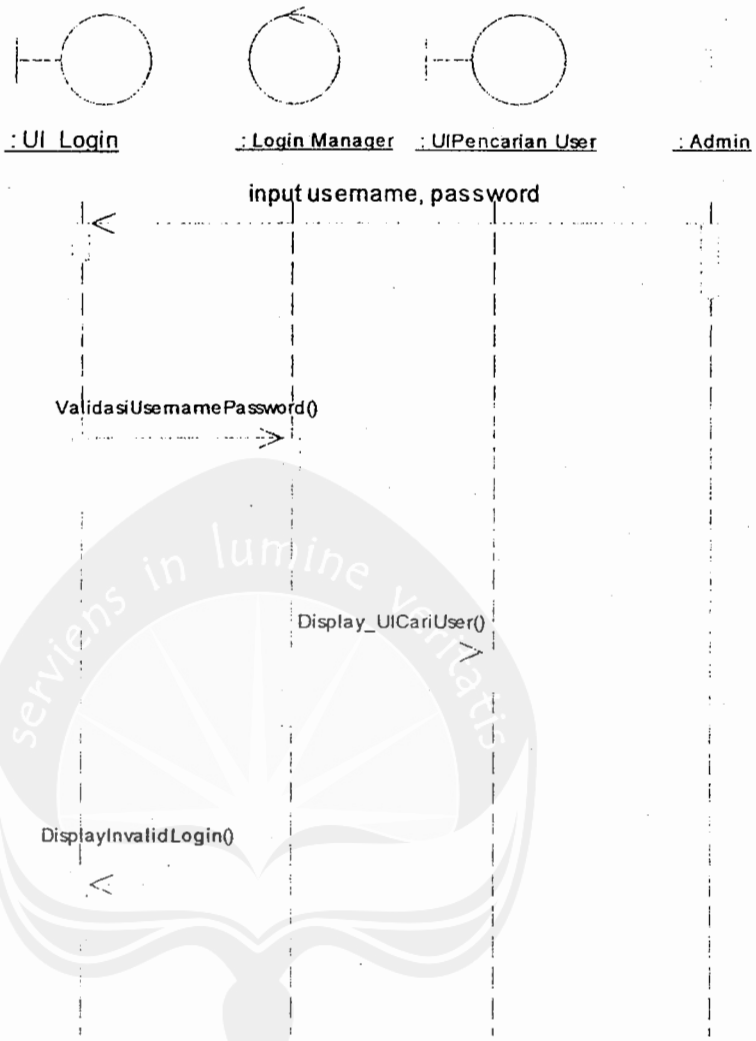
2.38 Use Case : Pengelolaan Barang

Flow of events:

1. Administrator memasukkan data barang yang baru melalui `class boundary UIPenambahanBarang`
2. Admin memicu terjadinya pembentukan objek `Barang` yang merupakan *instance* dari `class Barang`
3. Objek `Barang` terbentuk
4. Class `ManagerPengelolaanBarang` Memanggil method `getBarang()` yang terdapat di class `Barang` untuk mengambil data barang
5. `Class ManagerPengelolaanBarang` Menambahkan data barang ke dalam database
6. `Class boundary UIPenambahanBarang` memanggil `method getStatusAddBarang()` yang terdapat didalam `class ManagerPengelolaanBarang`, untuk mengetahui apakah proses penambahan berhasil atau gagal
7. `Class boundary UIPenambahanBarang` menampilkan pesan konfirmasi penambahan barang
8. Administrator memasukkan data barang yang baru melalui `class boundary UI_UpdateBarang`
9. Admin memicu terjadinya pembentukan objek `Barang` yang merupakan *instance* dari `class Barang`
10. Objek `Barang` terbentuk
11. Class `ManagerPengelolaanBarang` Memanggil method `getBarang()` yang terdapat di class `Barang` untuk mengambil data barang
12. `Class ManagerPengelolaanBarang` Mengupdate data barang
13. `Class UI_UpdateBarang` memanggil method `getStatusUpdate()` untuk mengetahui apakah proses update data barang sukses atau gagal

14. `Class boundary UI_UpdateBarang` menampilkan pesan konfirmasi update barang
15. Administrator memasukkan data barang yang baru melalui `class boundary UI_HapusBarang`
16. Admin memicu terjadinya pembentukan objek `Barang` yang merupakan *instance* dari `class Barang`
17. Objek `Barang` terbentuk
18. Class `ManagerPengelolaanBarang` Memanggil method `getBarang()` yang terdapat di class `Barang` untuk mengambil data barang
19. `Class ManagerPengelolaanBarang` Menghapus data barang
20. `Class UI_HapusBarang` memanggil method `getStatusHapus()` untuk mengetahui apakah proses delete data barang sukses atau gagal
21. `Class boundary UI_HapusBarang` menampilkan pesan konfirmasi delete barang
22. Administrator memasukkan data barang yang baru melalui `class boundary UI_CariBarang`
23. Admin memicu terjadinya pembentukan objek `Barang` yang merupakan *instance* dari `class Barang`
24. Objek `Barang` terbentuk
25. `Class ManagerPengelolaanBarang` Memanggil `method getBarang()` yang terdapat di `class Barang` untuk mengambil data barang
26. `Class UI_CariBarang` memanggil `method CariBarang()` yang terdapat didalam `class ManagerPengelolaanBarang` untuk melakukan pencarian barang
27. `Class UICariBarang` Menampilkan data barang

2.3.7. Use Case : Login



2.39 Use Case : Login

Flow of Events:

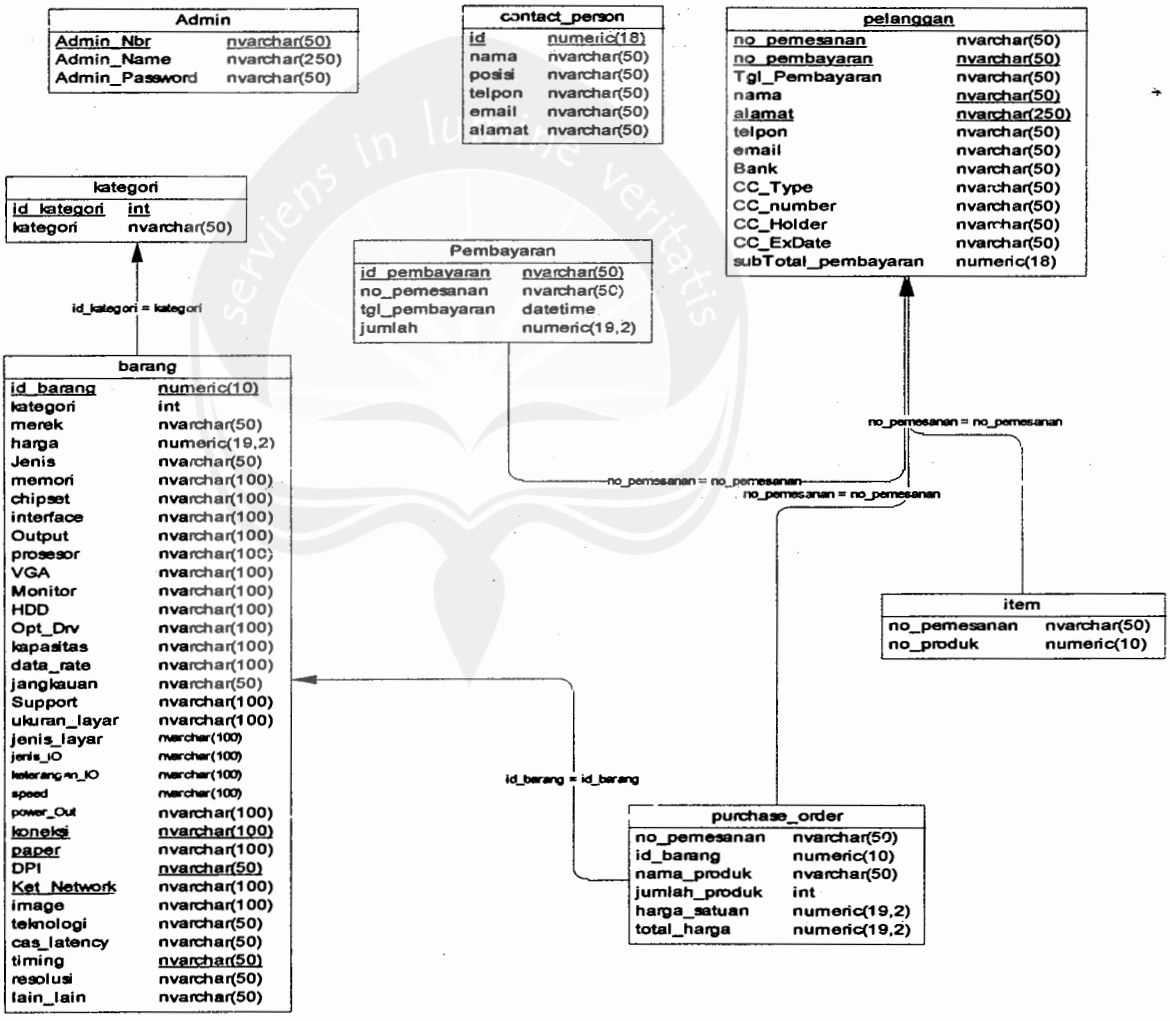
1. Admin memasukkan username dan password melalui `class boundary UI_Login`
2. Class UI_Login memanggil method `ValidasiUsernamePassword()` mengetahui apakah username dan password benar
3. Jika username dan password benar maka `class LoginManager` akan memanggil method

Display_UICariUser() untuk menampilkan antarmuka pencarian user sebagai awalan unutk melakukan pengelolaan user maupun pengelolaan barang

4. Jika usename dan password salah maka `class UI_Login` akan menjalankan method `DisplayInvalidLogin()` sehingga muncul peringatan bahwa username dan password salah

3. Deskripsi Perancangan Persistent Data

3.1. Database



Gambar 3.1 Physical Diagram ARKAJAX

Telepon	Numeric	10	Nomor telpon pelanggan yang bisa dihubungi
Email	Char	20	*Email* pelanggan
Kartu Kredit	Numeric	11	Kartu kredit yang dimiliki pelanggan

Tabel 3.1 Tabel Pelanggan

3.1.2. Tabel Memesan

Atribut	Tipe data	Ukuran	Deskripsi
No_pemesanan	Numeric	10	*Primary key* pelanggan
Kode_Produk	Numeric	10	*Primary key* produk
Jumlah_produk	Numeric	10	Jumlah produk yang dipesan oleh pelanggan
Total_harga	Money		Total harga pemesanan
Nama_produk	Char	20	Merek produk yang dipesan
Harga_satuan	Money		Harga produk per unit

Tabel 3.2 Tabel Memesan

3.1.3. Tabel Pembayaran

Atribut	Tipe data	Ukuran	Deskripsi
kode_pembayaran	Numeric	10	Kode pembayaran
tgl_pembayaran	Date		Tanggal pembayaran
Bank	Char	20	Bank yang digunakan sebagai perantara pembayaran
Credit_card_owner	Char	20	Nama pemilik kartu kredit
Credit_Card_nmbr	Numeric	15	Nomor kartu kredit
Amount	Money		Total pemesanan

			yang harus dibayar
Metode_pembayaran	Char	10	Cara pembayaran yang dilakukan oleh pelanggan, apakah dengan kartu kredit atau melalui rekening bank

Tabel 3.3 Tabel Pembayaran

3.1.4. Tabel Item

Atribut	Tipe data	Ukuran	Deskripsi
Kode_Produk	Numeric	10	Kode item yang dipesan
No_pemesanan	Numeric	10	Kode pemesanan

Tabel 3.4 Tabel Item

3.1.5. Tabel Produk

Atribut	Tipe data	Ukuran	Deskripsi
Kode_Produk	Numeric	10	*Primary key* produk
Nama	Char	20	Nama produk
Jenis	Char	20	Jenis produk
Harga	Money		Harga produk

Tabel 3.5 Tabel Produk

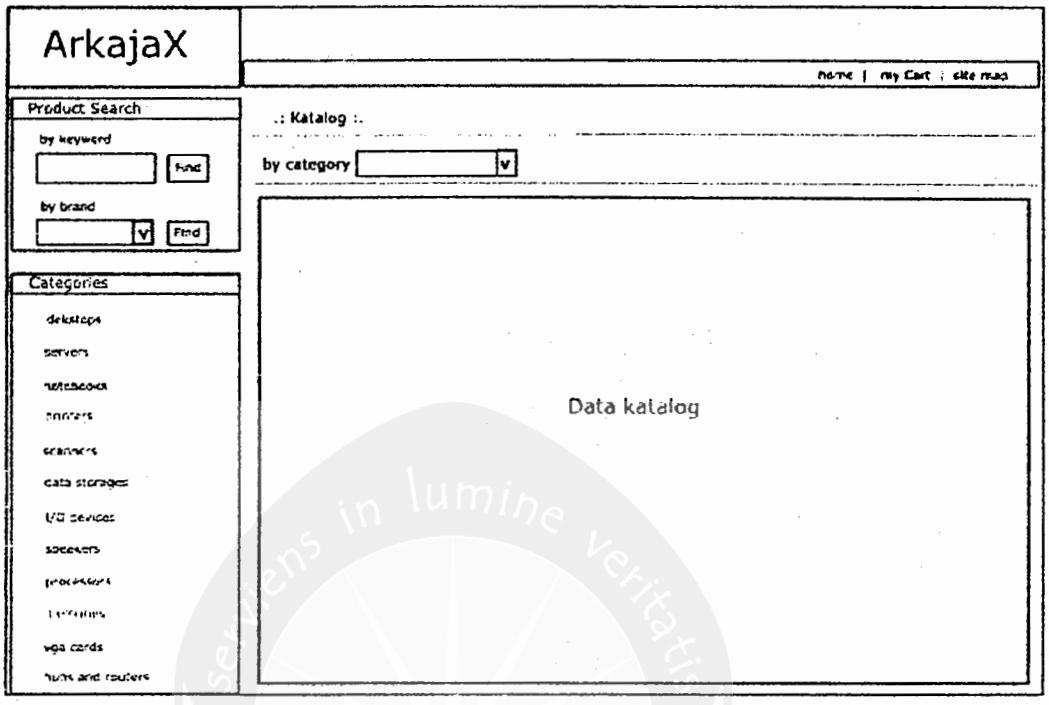
3.1.6. Tabel Admin

Atribut	Tipe data	Ukuran	Deskripsi
No_Admin	Char	10	Primary key Admin
Nama_Admin	Char	50	Nama administrator
Password	Char	50	Password admin

Tabel 3.6 Tabel Admin

4. Deskripsi Perancangan Antarmuka

4.1. Use Case : *Browsing* Katalog



Gambar 4.1 Antarmuka *Browsing* Katalog

Deskripsi :

- Antarmuka ini diimplementasikan pada *class* UIKatalog, antar muka ini digunakan pada *use case Browsing* Katalog (UC-Brow-ARK01). Antarmuka ini digunakan oleh *user* untuk melakukan *browsing* katalog penjualan komputer
- *ComboBox* di halaman utama digunakan untuk mencari katalog berdasarkan merek produk berdasarkan kategori tertentu yang terdapat pada Menu *categories* di kiri bawah
- Menu *categories* yang terdapat di kiri bawah digunakan untuk mencari produk berdasarkan kategori produk
- *Combobox by brand* digunakan untuk mencari produk berdasarkan merek produk
- *Textbox by keyword* digunakan untuk mencari katalog penjualan komputer yang kata kuncinya ditentukan oleh *user* sendiri.

Events :

- Mencari katalog berdasarkan kategori

  Untuk dapat melakukan *event* ini, *user* dapat memilih menu kategori produk yang terdapat pada *panel categories*

  Urutan aksi yang terjadi :

  1. *User* memilih menu yang terdapat pada *panel categories*
  2. `UIKatalog` menampilkan katalog produk yang dijual jika dan hanya jika memiliki kategori yang nilainya sesuai dengan menu kategori yang dipilih *user*.

  Algoritma (Menu_kategori_onclick) :

  1. input : Menu_kategori_value
  2. Sistem mencari katalog didalam database, perintah SQLnya sebagai berikut:

     ```
     "SELECT * FROM produk WHERE
     kategori_produk=Menu_kategori_value"
     ```

  3. Output (tampilkan katalog berdasarkan kategori)

- Mencari katalog berdasarkan *keyword*

  Untuk dapat melakukan *event* ini, *user* menginputkan kata kunci pencarian katalog yang terdapat pada panel *product search*, kemudian mengklik tombol *"Find"*

  Urutan aksi yang terjadi :

  1. *User* memasukkan kata kunci pada *textbox keyword*
  2. *User* menekan tombol *Find*
  3. UIKatalog menampilkan semua data yang sesuai dengan kata kunci yang diinputkan oleh *user*

  Algoritma (btn_find_onclick) :

  1. input : txtbox_keyword
  2. Sistem mencari katalog didalam database, perintah SQLnya sebagai berikut:

```
"SELECT * FROM produk WHERE
kode_produk=txtbox_keyword OR
nama=txtbox_keyword OR jenis=txtbox_keyword
OR harga=txtbox_keyword OR
kategori_produk=txtbox_keyword"
```

3. Output (tampilkan katalog berdasarkan keyword)

- Mencari katalog berdasarkan merek produk

  Untuk dapat melakukan *event* ini, *user* dapat memilih merek produk di *combobox by brand* yang terdapat di dalam *panel product search*

  Urutan aksi yang terjadi :

  1. *User* memilih *item* di combobox *by brand* yang terdapat di dalam panel *product search*
  2. `UIKatalog` menampilkan katalog produk yang dijual jika dan hanya jika memiliki merek yang nilainya sesuai dengan *item* yang terdapat didalam *combobox*.

  Algoritma (Btn_find_onclick) :

  1. input : Menu_brand_value
  2. Sistem mencari katalog didalam database, perintah SQLnya sebagai berikut:

     ```
     "SELECT * FROM produk WHERE
     Nama=Menu_brand_value"
     ```

  3. Output (tampilkan katalog berdasarkan merek)

- Mencari katalog berdasarkan kategori dan merek produk

  Untuk dapat melakukan *event* ini, *user* dapat memilih menu kategori produk yang terdapat pada panel *categories*, kemudian *user* menentukan merek produk yang kategorinya telah ditentukan oleh *user*

  Urutan aksi yang terjadi :

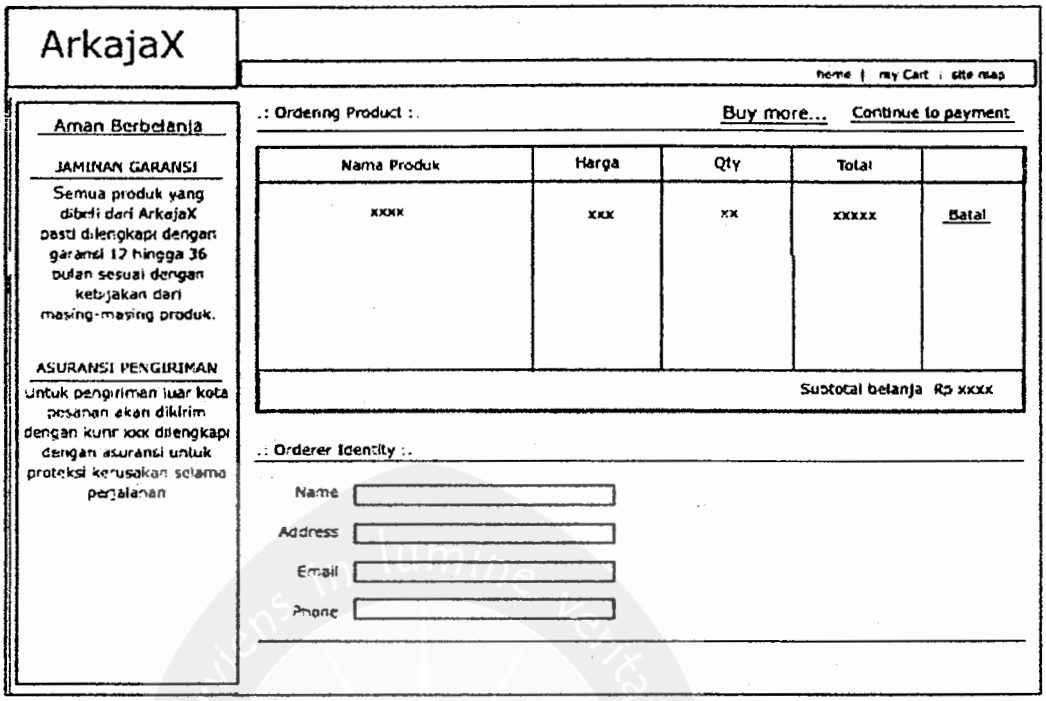
  1. *User* memilih menu yang terdapat pada *panel categories*

2. `UIKatalog` menampilkan katalog produk yang dijual jika dan hanya jika memiliki kategori yang nilainya sesuai dengan menu kategori yang dipilih *user*.
3. *User* memilih *combobox by brand* yang terdapat dihalaman utama
4. UIkatalog menampilkan katalog produk sesuai dengan kategori dan merek yang dipilih

Algoritma (Menu_kategori_onclick) :

1. input : Menu_kategori_value
2. Sistem mencari katalog didalam database, perintah SQLnya sebagai berikut:
   *"SELECT * FROM produk WHERE kategori_produk=Menu_kategori_value"*
3. Output (tampilkan katalog berdasarkan kategori)
4. input : Menu_brand_value
5. Sistem mencari katalog berdasarkan kategori dan mereknya didalam database, perintah SQLnya sebagai berikut:
   *"SELECT * FROM produk WHERE kategori_produk=Menu_kategori_value AND Nama=Menu_brand_value"*
6. Output (tampilkan katalog berdasarkan kategori dan mereknya)

4.2. Use Case : Memesan Komputer



Gambar 4.2 Antarmuka Memesan Komputer

Deskripsi

- Antarmuka ini diimplementasikan pada class UIPemesanan, antarmuka ini digunakan pada *use case* Memesan Komputer (UC-Psn-ARK02). Antarmuka ini digunakan oleh *user* untuk melakukan pemesanan produk

Events

- Pembatalan item yang telah masuk dalam daftar pemesanan

  User dapat melakukan pembatalan terhadap *item* yang telah masuk ke daftar pemesanan dengan cara menekan tombol batal yang terdapat di daftar pemesanan

  Urutan aksi yang terjadi :

  1. *User* menekan tombol "batal"
  2. UIPemesanan menghapus *item* yang dipesan

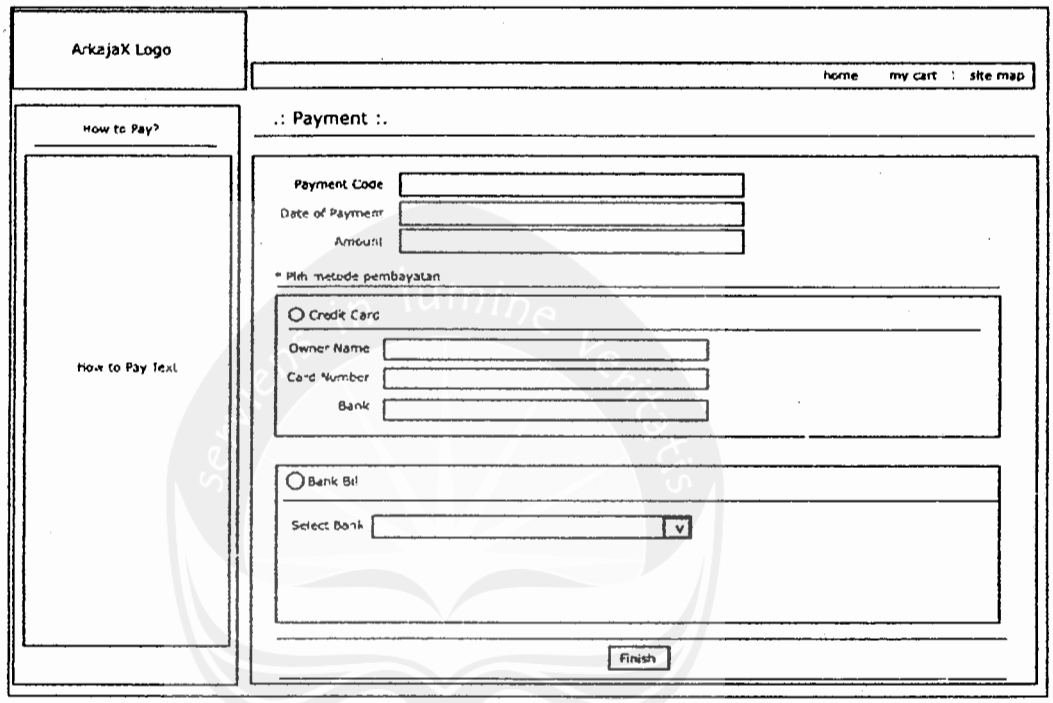
  Algoritma (btn_batal_onclick) :

  1. input : btn_batal_value

2. Sistem menghapus *item* terpilih yang terdapat didalam *database* perintah SQLnya sebagai berikut :

```
"DELETE FROM Memesan where merek=
btn_batal_value"
```

4.3. Use Case : Membayar Pemesanan Komputer



Gambar 4.3 Antarmuka membayar pemesanan komputer

Deskripsi

- Antarmuka ini diimplementasikan pada *class* UIPembayaran, antarmuka ini digunakan pada *use case* Membayar Pemesanan Komputer (UC-Byr-ARK03). Antarmuka ini digunakan oleh *user* untuk melakukan pembayaran produk yang telah dipesan.

Events

- Membayar dengan kartu kredit

  Untuk dapat melakukan *event* ini *radio button "Credit Card"* harus dipilih. Hal ini mengakibatkan komponen-komponen pada *panel*

"*Credit Card*" menjadi aktif dan dapat diedit, sehingga memasukan *detail* kartu kredit pada tempat yang tersedia.

Urutan aksi yang terjadi :

1. *User* memasukkan nilai variabel-variabel pada *panel "Credit Card"*.
2. *User* Menekan tombol *"Finish"*
3. `UIPembayaran` akan membentuk *instance* baru *class* `DataPembayaran`

Algoritma :

1. Input : Text_Ccard_Owner _value
2. Input : Text_Ccard_number_value
3. Input : Text_Bank_value
4. Input : Text_payment_code_value
5. Input : Text_payment_date_value
6. Input : Text_amount_value
7. Sistem menyimpan data pembayaran ke dalam database perintah SQLnya, sebagai berikut :

```
"INSERT INTO Pembayaran (Credit_card_owner,
Credit_card_nmbr, Bank, kode_pembayaran,
tgl_pembayaran, amount, metode_pembayaran)
VALUES (Text_Ccard_Owner _value,
Text_Ccard_number_value, Text_Bank_value,
Text_payment_code_value,
Text_payment_date_value, Text_amount_value,
'CC')"
```

- Membayar dengan rekening *bank*

  Untuk dapat melakukan *event* ini *radio button "Bank Bill"* harus dipilih. Hal ini mengakibatkan komponen-komponen pada *panel* "*Bank Bill*" menjadi aktif dan dapat diedit, sehingga memasukan *detail* kartu kredit pada tempat yang tersedia

Urutan aksi yang terjadi :

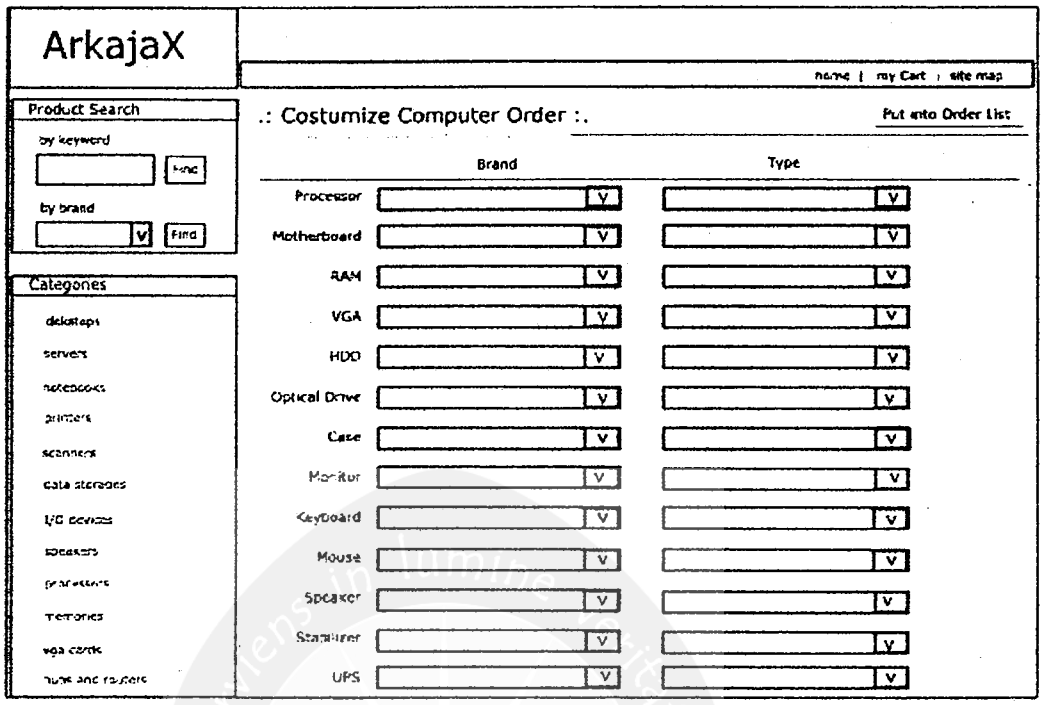
1. *User* memasukkan nilai variabel-variabel pada *panel "Bank Bill"*.
2. *User* Menekan tombol *"Finish"*
3. `UIPembayaran` akan membentuk *instance* baru *class* `DataPembayaran`

Algoritma :

1. Input : Combobox_Bank_value
2. Input : Text_payment_code_value
3. Input : Text_payment_date_value
4. Input : Text_amount_value
5. Sistem menyimpan data pembayaran ke dalam database perintah SQLnya, sebagai berikut :

```
"INSERT INTO Pembayaran (Bank,
kode_pembayaran, tgl_pembayaran, amount,
metode_pembayaran) VALUES
(Combobox_Bank_value,
Text_payment_code_value,
Text_payment_date_value, Text_amount_value,
'Rek')"
```

4.4. Use Case : Membuat Spesifikasi Komputer



Gambar 4.4 Antarmuka membayar pemesanan computer

Deskripsi

Antarmuka ini diimplementasikan pada *class* UIPembayaran, antarmuka ini digunakan pada *use case* Membayar Pemesanan Komputer (UC-Byr-ARK03). Antarmuka ini digunakan oleh *user* untuk melakukan pembayaran produk yang telah dipesan.

Events

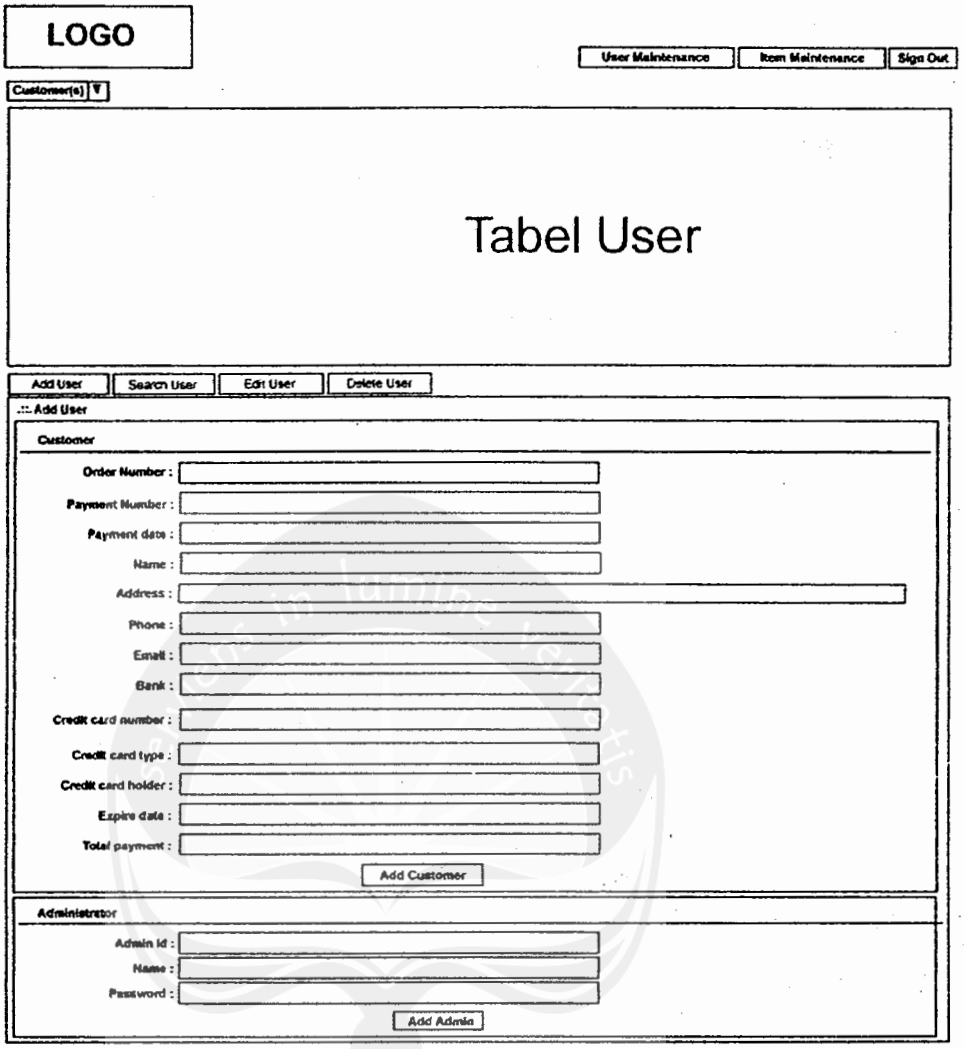
- Memindahkan daftar spesifikasi komputer ke daftar pemesanan

  Untuk dapat melakukan *event* ini, *user* harus memilih terlebih dahulu variabel spesifikasi komputer yang akan dipesannya.

  Urutan aksi yang terjadi :

  1. *User* memasukkan variabel spesifikasi komputer yang akan dipesan
  2. *User* menekan tombol "*Put into Order List*"

4.5. Use Case : Pengelolaan User



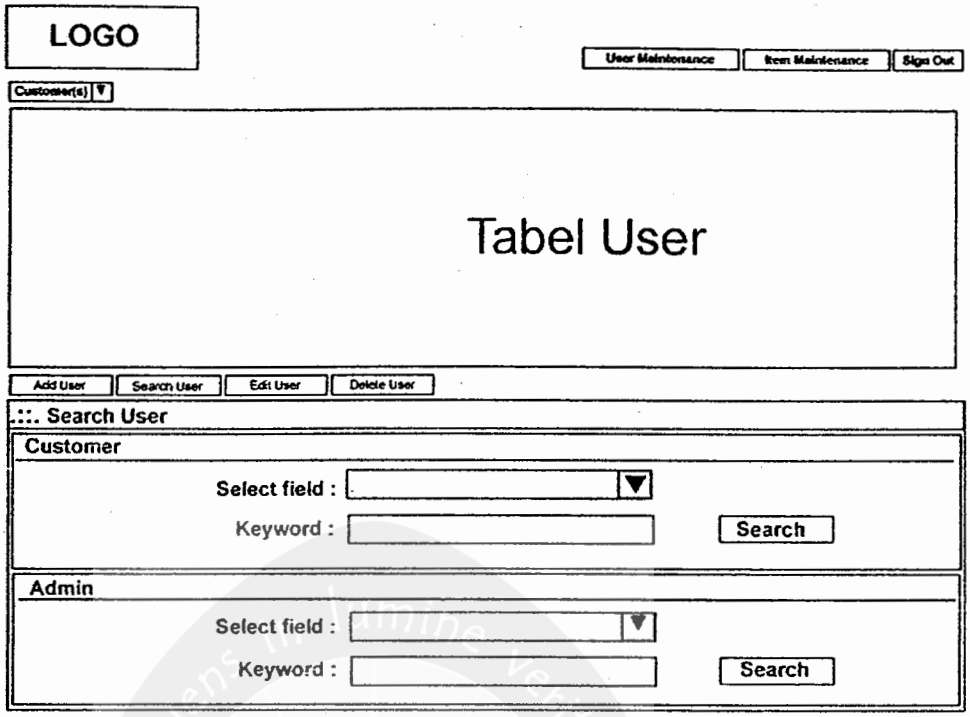
Gambar 4.5 Antarmuka penambahan data user

Deskripsi

Antarmuka ini dimplementasikan pada `class UIPenambahanUser`, antarmuka ini digunakan pada *use case* Pengelolaan User (UC-User-ARK05). Antarmuka ini digunakan oleh admin untuk melakukan penambahan data user.

Events:

1. Admin memasukkan data user yang baru
2. Admin menekan tombol "*Add User*"



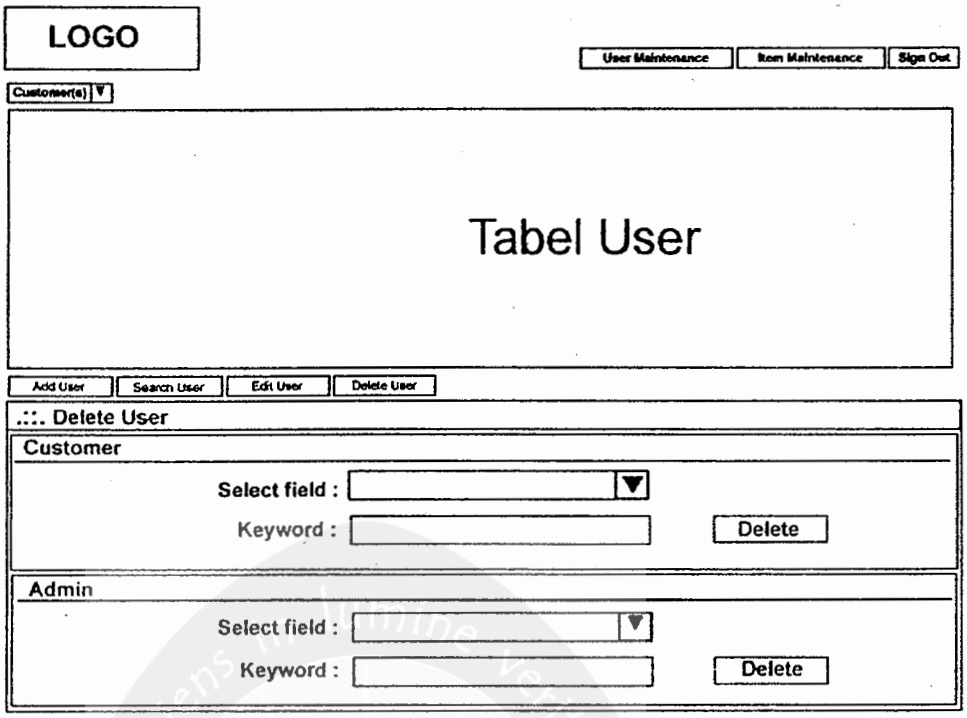
Gambar 4.6 Antarmuka Untuk Mencari User

Deskripsi

Antarmuka ini dimplementasikan pada `class UIPencarianUser`, antarmuka ini digunakan pada *use case* Pengelolaan User (UC-User-ARK05). Antarmuka ini digunakan oleh admin untuk melakukan pencarian data user.

Events:

1. Admin menentukan field yang akan digunakan untuk mencari data use
2. Admin memasukkan data user ke dalam textbox
3. Admin menekan tombol "*Search User*"



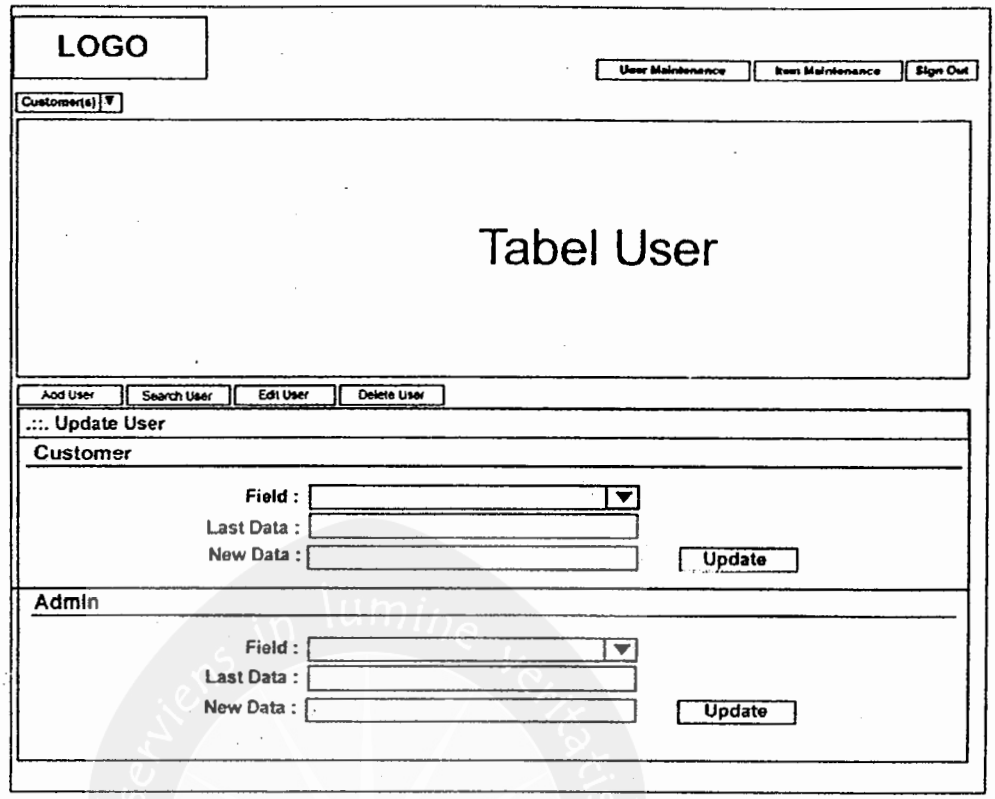
Gambar 4.7 Antarmuka Untuk Menghapus User

Deskripsi

Antarmuka ini dimplementasikan pada `class UIHapusUser`, antarmuka ini digunakan pada *use case* Pengelolaan User (UC-User-ARK05). Antarmuka ini digunakan oleh admin untuk melakukan penghapusan data user.

Events:

1. Admin menentukan field data yang akan dihapus
2. Admin memasukkan data user ke dalam textbox
3. Admin menekan tombol "*Delete*"



Gambar 4.8 Antarmuka Untuk Men-update User

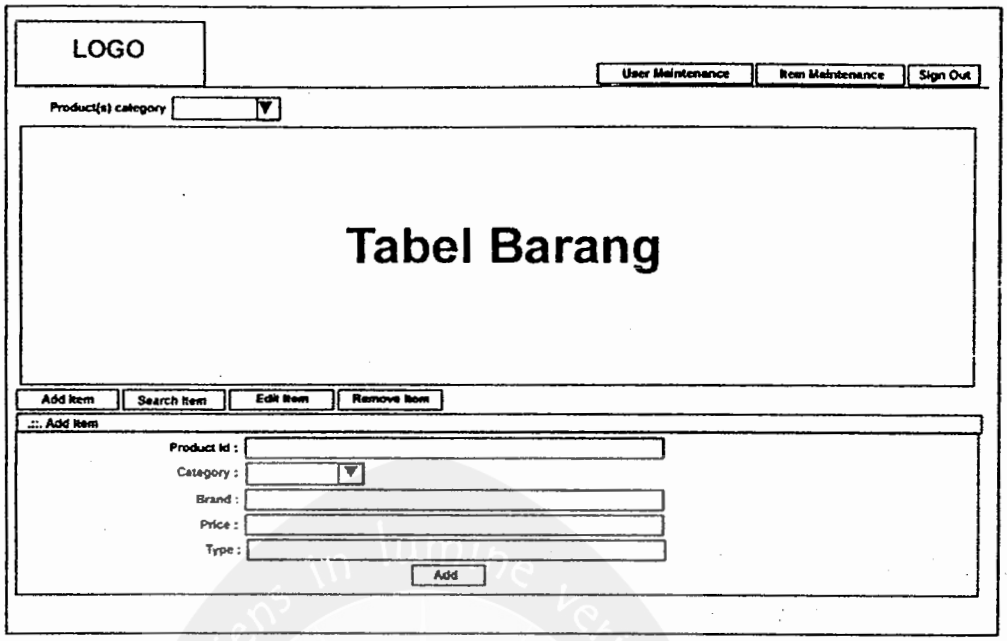
Deskripsi

Antarmuka ini dimplementasikan pada `class UI_UpdateUser`, antarmuka ini digunakan pada *use case* Pengelolaan User (UC-User-ARK05). Antarmuka ini digunakan oleh admin untuk melakukan update data user.

Events:

1. Admin menentukan field yang terdapat didalam textbox yang akan diupdate nilainya
2. Admin memasukkan data lama yang akan diupdate kedalam textbox
3. Admin memasukkan data baru yang akan diupdate kedalam textbox
4. Admin menekan tombol *"update"*

4.6. Use Case : Pengelolaan Barang



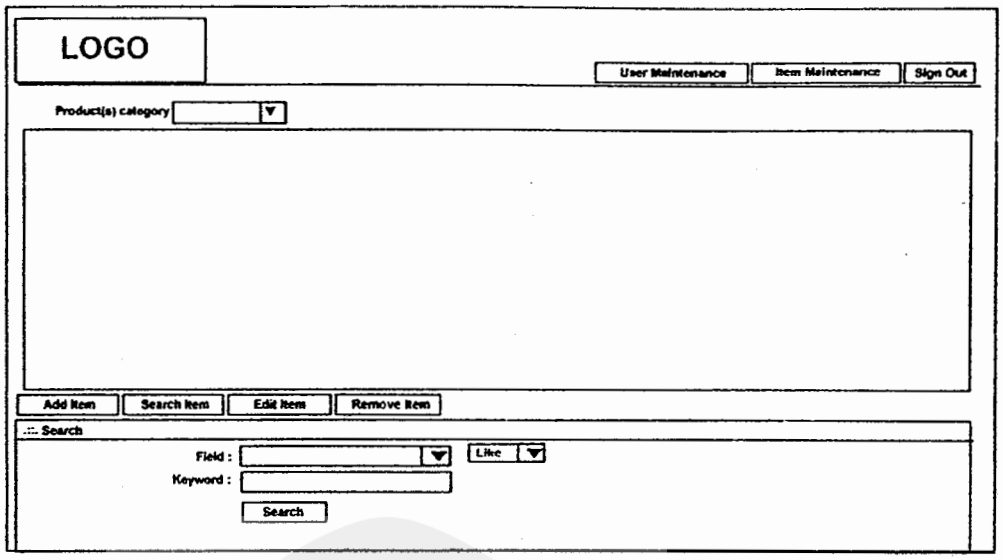
Gambar 4.9 Antarmuka Untuk Menambahkan Data Barang

Deskripsi

Antarmuka ini dimplementasikan pada `class UI_PenambahanBarang`, antarmuka ini digunakan pada *use case* Pengelolaan Barang (UC-Barang-ARK06). Antarmuka ini digunakan oleh admin untuk melakukan penambahan data barang.

Events:

1. Admin memasukkan data user kedalam textboxt yang tersedia
2. Admin menekan tombol "*Add*"



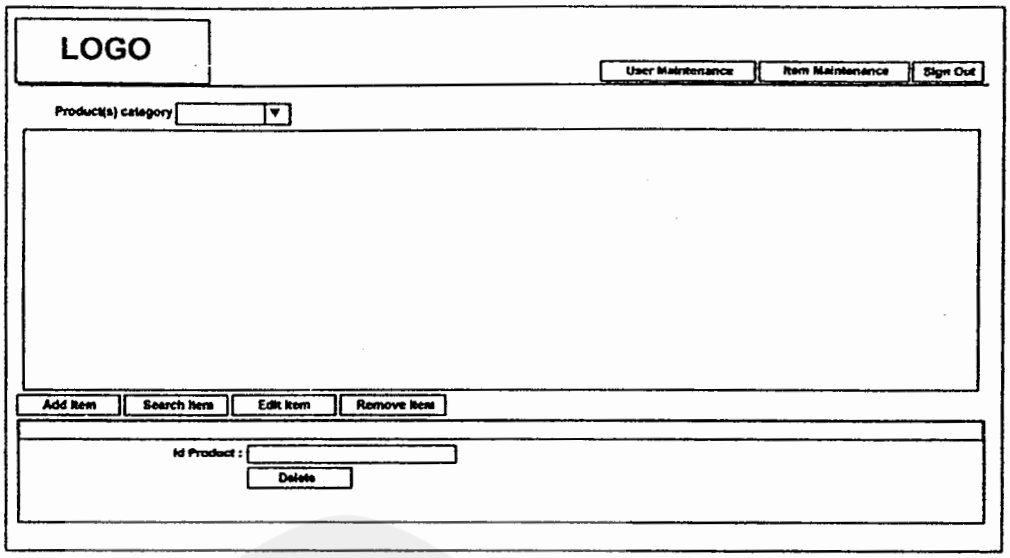
Gambar 4.10 Antarmuka Untuk Mencari Data Barang

Deskripsi

Antarmuka ini dimplementasikan pada `class UI_CariBarang`, antarmuka ini digunakan pada *use case* Pengelolaan Barang (UC-Barang-ARK06). Antarmuka ini digunakan oleh admin untuk melakukan pencarian data barang.

Events:

1. Admin menentukan field yang akan digunakan untuk mencari data barang
2. Admin memasukkan data barang ke dalam textbox
3. Admin menekan tombol "*Search*"



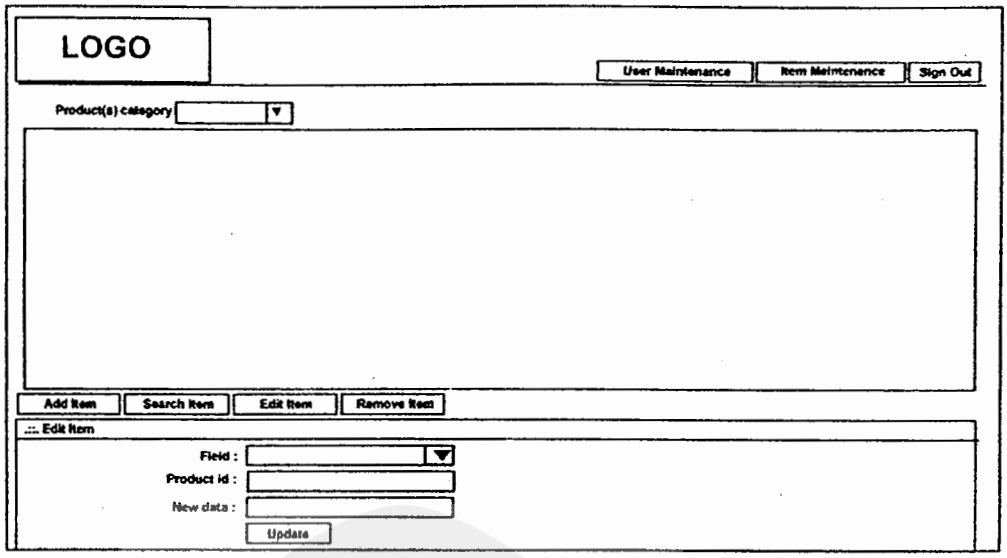
Gambar 4.11 Antarmuka Untuk Menghapus Barang

Deskripsi

Antarmuka ini dimplementasikan pada `class UI_HapusBarang`, antarmuka ini digunakan pada *use case* Pengelolaan Barang (UC-Barang-ARK06). Antarmuka ini digunakan oleh admin untuk melakukan penghapusan data barang.

Events:

1. Admin memasukkan id barng yang akan dihapus kedalam textbox
2. admin menekan tombol *"delete"*



Gambar 4.12 Antarmuka Untuk Men-edit Data Barang

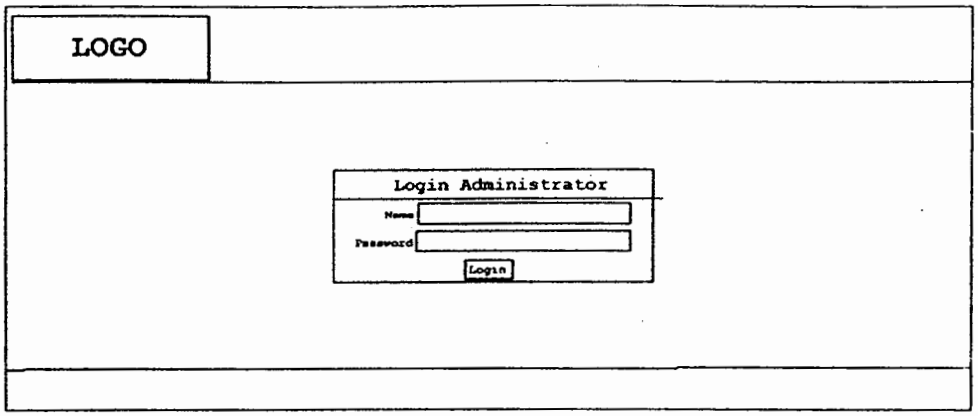
Deskripsi

Antarmuka ini dimplementasikan pada `class UI_UpdateBarang`, antarmuka ini digunakan pada *use case* Pengelolaan Barang (UC-Barang-ARK06). Antarmuka ini digunakan oleh admin untuk melakukan update data barang.

Events:

1. Admin menentukan field yang akan diupdate
2. Admin memasukkan id barang yang akan diupdate ke dalam textbox
3. Admin memasukkan data yang akan update kedalam textbox
4. Admin menekan tombol *"Update Item"*

4.7. Use Case : Login



Gambar 4.13 Antarmuka Untuk Proses Login

Deskripsi

Antarmuka ini dimplementasikan pada `class UI_Login`, antarmuka ini digunakan pada *use case* Login (UC-Login-ARK07). Antarmuka ini digunakan oleh admin untuk login sebelum menggunakan `Use Case:Pengelolaan User` dan `Use Case:Pengelolaan Barang`.

Events:

1. Admin memasukkan usename kedalam textbox *"username"*
2. Admin memasukkan password kedalam textbox *"password"*
3. Admin menekan tombol *"login"*

Apendiks A : Daftar Istilah dan Singkatan

User Pengguna sistem ARKAJAX

Node adalah objek fisik yang merepresentasikan sumber daya untuk pemrosesan, yang biasanya memiliki memori dan kemampuan pemrosesan. Node dapat berupa alat komputasi, manusia, atau sumber daya pemrosesan mekanik lain

Client adalah pelanggan web retail ARKAJAX yang melakukan pemesanan produk

Server adalah komputer yang digunakan untuk menyimpan data

Web browser perangkat lunak untuk menjelajahi web retail ARKAJAX

Produk barang yang dijual dalam sistem ARKAJAX

Retail suatu kelompok/ organisasi yang bergerak di bidang perdagangan

SQL server perangkat lunak database relasional buatan microsoft, yang digunakan untuk menyimpan data pemesanan dan pembayaran

Package mekanisme pengelompokan yang digunakan untuk menandakan pengelompokan elemen-elemen model

Stereotype merupakan sub klasifikasi dari sebuah elemen model

Boundary kelas yang memodelkan interaksi antara satu atau lebih aktor dengan sistem

Control digunakan untuk memodelkan "perilaku mengatur" khusus untuk satu atau beberapa use case saja

Use Case merupakan diagram yang menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (aktor)

Entity memodelkan informasi yang harus disimpan oleh sistem

Persistent storage adalah objek fisik yang merepresentasikan sumber daya untuk pemrosesan, yang biasanya memiliki memori dan kemampuan pemrosesan. Node dapat berupa alat komputasi, manusia, atau sumber daya pemrosesan mekanik lain

Atribut merupakan property sebuah objek

Item merupakan barang yang dibeli oleh pelanggan sistem ARKAJAX

Database kumpulan file/arsip/tabel yang disimpan dalam media elektronis dan saling berhubungan

Katalog daftar produk yang dijual di web retail ARKAJAX

Rekening nomor identifikasi kepemilikan tabungan di sebuah bank

Kartu kredit kartu yang digunakan sebagai alat pembayaran non-cash

Valid keabsahan suatu benda

Flow of event adalah aliran event-event yang terjadi dalam suatu proses, yang digunakan untuk mendeskripsikan urutan jalannya proses.

Validasi memeriksa keabsahan suatu benda