Kisi-Kisi UTS 2024

- nomer 1 terkait SRS
- nomer 2 dan 3 terkait Usecase diagram dan activity diagram
- nomer 4 membuat Usecase diagram dan activity diagram berdasarkan studi kasus
- open hard file (no gadget, no laptop)

SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS)

Definisi "Requirements"

spesifikasi tentang **apa yang harus diimplementasikan** dalam sebuah sistem. Mereka merupakan deskripsi tentang **bagaimana sistem seharusnya berperilaku**, atau tentang **properti atau atribut sistem**. Persyaratan juga dapat berupa ****batasan pada proses pengembangan sistem**.

Sesuatu yang harus dilakukan dan harus dimiliki untuk meningkatkan kualitas suatu produk

Sebuah spesifikasi yang harus diimplementasikan, yaitu deskripsi sistem, properti sistem, atribut sistem, yang bisa jadi kendala pada proses pengembangan sistem. Rekayasa sistem suatu produk menuntut fungsi dan kualitas, karena klien menginginkan rekayasa sistem menjadi bagian dari produk yang mereka pesan

Istilah SRS

Kontrak: dokumen yang mengikat secara hukum dan disepakati oleh klien dan developer, termasuk syarat-syarat teknologi dan organisasi, biaya, serta jadwal pengerjaan. Kontrak bisa mengandung sesuatu yang kurang formal tetapi bermanfaat, seperti komitmen atau harapan dari pihak yang terlibat.

Klien/mitra : Pihak yang membayar untuk produk dan biasanya yang menentukan persyaratan (requirements).

Developer/pengembang IT: Pihak yang membuat produk software untuk klien.

Pengguna: Pihak yang mengoperasikan atau berinteraksi langsung dengan software. Pengguna dan klien biasanya bukan orang yang sama.

Manfaat SRS

Sebagai bentuk perjanjian antara klien dan developer tentang software apa yang akan dibuat.

Mengurangi beban dalam proses pengembangan software.

Sebagai bahan perkiraan biaya dan rencana penjadwalan.

Sebagai dasar validasi dan verifikasi software di ujung penyelesaian proyek nantinya.

Memfasilitasi transfer, semisal software tersebut ingin ditransfer ke pengguna atau mesin-mesin yang lain. Customer pun merasa mudah jika ingin mentransfer software ke bagian-bagian lain dalam organisasinya. Bahkan, jika terjadi pergantian personil developer, proyek dapat mudah ditransfer ke personil baru dengan memahami SRS ini.

Mendasari perbaikan produk software di kemudian hari. Jadi, kadang SRS boleh diperbaiki dengan alasan dan mekanisme tertentu serta atas kesepakatan antara customer dan developer.

Ruang lingkup

Mengingat SKPL pada akhirnya akan menjadi dasar bagi kontrak antara developer dan customer , maka suatu dokumen SKPL harus memenui syarat-syarat berikut:

1. Mendefinisikan kebutuhan software dengan benar. Kebutuhan software muncul karena ada pekerjaan yang harus diselesaikan atau karena ada karakteristik khusus dari proyek.
2. Tidak menjelaskan rancangan atau implementasi dengan rinci. Penjelasan tersebut tidak diperlukan karena bagi pengguna hal tersebut lebih teknis dan tidak perlu.
3. Tidak memaksakan penambahan suatu batasan dari perangkat lunak

Karakteristik SRS

1. Correct (benar)
2. Unambiguous (tidak ambigu, tapi jelas)
3. Complete (lengkap)
4. Consistent (konsisten)
5. Ranked for importance and/or stability (prioritas penting dan atau stabilitas)
6. Verifiable (dapat diverifikasi)
7. Modifiable (bisa dimodifikasi)
8. Traceable (bisa dilacak)

Keterlibatan dalam Pembuatan SRS

1. Pemakai (user), kelompok orang yang mengoperasikan/menggunakan produk final dari system yang dibuat.
2. Client, orang atau perusahaan yang mau membuat sistem.
3. System Analyst (system engineer), kelompok orang yang biasa melakukan kontak Teknik pertama dengan client. Bertugas menganalisis persoalan, menerima requirement dan menulis requirement.
4. Software Engineer, kelompok orang yang bekerja setelah kebutuhan system dibuat (bekerjasama dengan system engineer saat mendefinisikan kebutuhan sistem dan membuat deskripsi perancangannya)
5. Programmer, kelompok orang yang menerima spesifikasi perancangana sistem, membuat kode dalambentuk modul, menguji dan memeriksa (tes) modul.
6. Test integration group, kelompok orang yang melakukan tes dan mengintegrasikan modul.
7. Maintenance group, kelompok orang yang memantau dan merawat performa sistem-sistem yang dibuat selama pelaksanaan dan pada saat modifikasi muncul (80% dari pekerjaan)
8. Technical Support, orang-orang yang mengelola (manage) pengembang sistem, termasuk konsultan atau orang yang mempunyai kepandaian lebih tinggi.
9. Staff dan Clerical Work, Kelompok orang yang bertugas mengetik, memasukkan data, dan membuat dokumen.

Petunjuk Penulisan SRS

Gunakan format yang standar dan gunakan pada seluruh kebutuhan yang dituliskan

Gunakan bahasa yang konsisten dan tidak membingungkan atau ambigu.

Gunakan kata "harus" pada kondisi kebutuhan yang memang harus dipenuhi oleh sistem, dan gunakan kata "seharusnya/sebaiknya" untuk menuliskan kondisi kebutuhan yang diinginkan.

Hindari penggunaan bahasa yang terlalu bersifat teknis dalam penulisan requirements.

Skema SRS

Persyaratan Fungsional langsung berhubungan dengan proses yang harus dikerjakan atau informasi yang harus dimuat oleh system.

Persyaratan fungsional merupakan fungsi dasar dari sebuah system. Dapat dikatakan, jika Persyaratan Fungsional dari system tidak berfungsi, maka hilanglah manfaat dari system

Pada Software Requirement Specifications, dikenal ada 2 skema persyaratan/kebutuhan, yaitu

1) Persyaratan Fungsional (Functional Requirement)

Menggambarkan fungsionalitas sistem atau layanan-layanan sistem

Sangat bergantung dari jenis perangkat lunak, pengguna sistem, dan jenis sistem dimana perangkat lunak tersebut digunakan

Kebutuhan fungsional dapat berupa pernyataan-pernyataan tingkat tinggi dari:

- Apa yang sistem harus lakukan
- Harus dapat menggambarkan layanan-layanan yang dapat diberikan oleh sistem kepada pengguna secara mendetail

Contoh "Sistem Perpustakaan"

Sistem perpustakaan menyediakan antarmuka tunggal untuk mengakses artikel-artikel dalam database perpustakaan yang berbeda-beda

Pengguna dapat mencari, men-download, dan mencetak artikel yang ditampilkan.

Contoh FR sistem perpustakaan:

- Pengguna harus dapat mencari di seluruh database yang ada, atau mencari di sebagian database yang disediakan.
- Sistem harus menyediakan program yang memungkinkan penggunanya membaca artikel atau dokumen dalam perpustakaan
- Sistem harus dapat mencatat buku-buku atau dokumen yang dipinjam oleh pengguna.

2) Persyaratan non-fungsional (Nonfunctional Requirement)

Non-functional requirements atau kebutuhan non-fungsional menentukan atribut atau kualitas secara keseluruhan dari suatu sistem.

Kebutuhan non-fungsional menempatkan batasan pada produk yang sedang dikembangkan, proses pengembangannya, dan menentukan batasan-batasan eksternal yang harus dipenuhi oleh produk tersebut.

Mengidentifikasi NFR

Umumnya NFR dapat dikenali dari "keinginan", "kebutuhan", atau "kepentingan" dari para *stakeholders*.

keinginan, kebutuhan, atau kepentingan tersebut umumnya bersifat non-fungsional

- Tujuan bisnis
- Karakteristik sistem
- Keamanan, performa, fungsionalitas, dan perawatan sistem

Kategori NFR

1) Keamanan(safety dan security)

Security

Kebutuhan akan keamanan disertakan dalam sistem untuk menjamin:

Akses ke dalam sistem tanpa otorisasi tidak dimungkinkan

Memastikan integritas sistem dari kecelakaan atau kerusakan

Contoh:

Data hanya dapat diubah oleh administrator sistem

Seluruh data harus di-backup setiap 24 jam, dan hasil backup-nya disimpan di lokasi yang berbeda dengan sistem

Seluruh komunikasi antara client-server harus dienkripsi

Safety

Umumnya diasumsikan sebagai seluruh kebutuhan yang terkait dengan keamanan sistem

Umumnya sangat berhubungan dengan kebutuhan untuk memastikan keamanan dalam operasional sistem

Kebutuhan untuk melindungi sistem

Kebutuhan untuk menghindari kecelakaan dalam sistem maupun dalam menggunakan sistem.

Penggunaannya seringkali bergantung dengan budaya dan perilaku dalam organisasi

Contoh:

Sistem tidak mengizinkan pengoperasian alat/perangkat kecuali terdapat petugas di lokasi

Sistem tidak boleh memberikan obat kepada pasien dengan dosis yang lebih dari yang diizinkan oleh dokter pasien yang bersangkutan

Sistem tidak boleh beroperasi jika suhu di luar ruangan berada di bawah 4 derajat Celcius.

Petugas yang melaksanaan proses perawatan sistem harus menggunakan gelang anti-statik selama berada di dalam ruangan.

2) Ketergunaan (usability)

Terkait dengan penentuan antarmuka dan interaksi pengguna dengan sistem

User manual yang terstruktur, pesan kesalahan yang informatif, fasilitas pertolongan, dan antarmuka yang konsisten dapat meningkatkan kebutuhan *usability* ini.

3) Reliabilitas/Reliability (diandalkan)

Memberikan batasan perilaku sistem pada saat beroperasi

Availability – Ketersediaan sistem dalam memberikan layanan ketika diperlukan oleh pengguna

Tingkat kegagalan – Seberapa sering sistem gagal untuk dapat memberikan layanan yang diharapkan oleh pengguna

4) Performansi/Performance

Memberikan batasan mengenai kecepatan operasional sebuah sistem

Kebutuhan akan respon

Kebutuhan throughput (keluaran)

Kebutuhan akan pewaktuan (timing)

Jenis- Jenis NFR

13 NFR yang harus disertakan dalam sebuah dokumen spesifikasi kebutuhan perangkat lunak (SRS)

1) Performance
2) Interface/Usability
3) Operational
4) Resource
5) Verification
6) Acceptance
7) Documentation
8) Security
9) Portability
10) Quality
11) Reliability
12) Maintainability
13) Safety

*Red=critical system NFR

Hubungan antara kebutuhan user, kepentingan, dan NFR

Kebutuhan	Kepentingan	NFR
Fungsi	1. Kemudahan penggunaan 2. Otoritas Akses 3. Kemungkinan kegagalan sistem	1. Usability 2. Security 3. Reliability
Performansi	1. Penggunaan sumber daya 2. Verifikasi performa 3. Kemudahan antarmuka	4. Efficiency 5. Verifiability 6. Interoperability
Perubahan	1. Kemudahan perbaikan 2. Kemudahan perubahan 3. Kemudahan transport/lokasi penggunaan 4. Kemudahan dalam meningkatkan kapasitas dan performa	5. Maintainability 6. Flexibility 7. Portability 8. Expandability

Contoh NFR

Contoh 1

"Sistem harus dapat memastikan bahwa data yang digunakan dalam sistem harus terlindung dari akses yang tidak berwenang."

Dengan kata lain, data tidak dapat diakses oleh pengguna yang tidak berhak.

Secara konvensional kebutuhan ini termasuk kebutuhan non-fungsional, karena tidak menyebutkan secara spesifik kebutuhan fungsional yang harus disediakan oleh sistem.

Namun dapat dispesifikasikan lebih lanjut dengan:

"Sistem harus menyertakan sebuah prosedur otorisasi dimana penggunanya harus mengidentifikasi diri dengan sebuah username dan password. Hanya pengguna yang memiliki wewenang melalui prosedur ini yang dapat mengakses data dalam sistem."

Contoh 2

Sistem layanan X harus memiliki tingkat ketersediaan 999/1000 atau 99%.

Hal ini merupakan kebutuhan akan reliabilitas yang berarti setiap 1000 permintaan layanan, 999 permintaan harus dapat terpenuhi.

Sistem Y harus mampu mengolah transaksi sekurang-kurangnya 8 transaksi dalam setiap detik.

Hal ini merupakan sebuah kebutuhan akan performansi.

Besarnya program sistem Z dibatasi sebesar 512 Kbytes.

Hal ini merupakan kebutuhan sumber daya (resource) yang menentukan ukuran memory maksimum sistem tersebut.

Contoh Penulisan Kebutuhan Fungsional

Contoh 1

1. Manajemen Kendaraan Baru

1.1 Sistem akan memungkinkan manajer untuk melihat inventaris kendaraan baru saat ini.

1.2 Sistem akan memungkinkan manajer kendaraan baru untuk melakukan pemesanan kendaraan baru.

1.3 Sistem akan mencatat penambahan kendaraan baru ke inventaris ketika diterima dari produsen.

2. Manajemen Penjualan Kendaraan

2.1 Sistem akan memungkinkan tenaga penjualan untuk membuat penawaran kepada pelanggan.

2.2 Sistem akan memungkinkan tenaga penjualan untuk mengetahui apakah ada penawaran yang tertunda pada kendaraan tertentu.

2.3 Sistem akan memungkinkan manajer untuk mencatat persetujuan penawaran pelanggan.

2.4 Sistem akan mempersiapkan kontrak penjualan.

2.5 Sistem akan mempersiapkan surat perintah kerja bengkel berdasarkan pilihan dealer yang diminta pelanggan.

2.6 Sistem akan mencatat deposit pelanggan.

2.7 Sistem akan mencatat pembayaran pelanggan.

2.8 Sistem akan membuat catatan pembelian kendaraan pelanggan.

3. Manajemen Kendaraan Bekas

3.1 Sistem akan mencatat informasi tentang kendaraan tukar tambah pelanggan... dll.

Contoh 2

Tabel 4 Kebutuhan Fungsional

ID	Kebutuhan (Needs)	Penjelasan (Explanation)
SRS-F-OUT-001	Perangkat lunak dapat menampilkan form sign in (The software can display a sign-in form)	Pengguna yang telah terdaftar dapat melakukan sign in sehingga perangkat lunak perlu menampilkan form sign in (Users who have registered can sign in, so the software needs to display a sign-in form)
SRS-F-OUT-002	Perangkat lunak dapat menampilkan form entri pendaftaran (The software can display a registration entry form)	Form entri pendaftaran ditampilkan saat pengguna melakukan pendaftaran sebagai pembeli (The registration entry form is displayed when the user registers as a buyer)
SRS-F-IN-003	Perangkat lunak dapat menerima masukan dari pengguna, berupa data pembeli (The software can accept input from the user, in the form of buyer data)	Data pembeli yang dimasukkan oleh pengguna dapat diterima oleh perangkat lunak (The buyer data entered by the user can be accepted by the software)
SRS-F-PR-004	Perangkat lunak dapat menyimpan data pembeli (The software can store buyer data)	Setelah pengguna yang melakukan pendaftaran sebagai pembeli memasukkan data yang dibutuhkan, perangkat lunak menyimpan data tersebut (After the user who registers as a buyer enters the required data, the software stores the data)
SRS-F-OUT-005	Perangkat lunak dapat menampilkan pesan berupa harga total yang harus dibayarkan (The software can display a message showing the total price to be paid)	Setelah perangkat lunak menghitung harga total yang harus dibayarkan pembeli, perangkat lunak menampilkan informasi tersebut (After the software calculates the total price to be paid by the buyer, the software displays this information)
SRS-F-PR-006	Perangkat lunak dapat menyediakan mode pencarian terhadap produk, sesuai dengan input dari pembeli (The software can provide a search mode for products, according to the buyer's input)	Untuk memilih produk, pengguna dapat menggunakan mode pencarian yang terdapat pada perangkat lunak (To choose a product, the user can use the search mode provided by the software)
SRS-F-IN-007	Perangkat lunak dapat menerima masukan username dan password pembeli (The software can accept username and password input from the buyer)	Username dan password yang dimasukkan pembeli harus dapat diterima oleh perangkat lunak (The username and password entered by the buyer must be accepted by the software)
SRS-F-OUT-008	Perangkat lunak dapat menampilkan katalog (The software can display a catalog)	Katalog memiliki informasi nama produk, kategori produk, harga produk, dan jumlah produk yang tersedia
SRS-F-PR-009	Perangkat lunak dapat menghitung jumlah uang yang harus dibayarkan oleh pembeli (The software can calculate the amount of money to be paid by the buyer)	Setelah pembeli memilih produk beserta jumlahnya, perangkat lunak melakukan pengitungan jumlah uang yang harus dibayarkan oleh pembeli
SRS-F-PR-010	Perangkat lunak dapat menyimpan data pemesanan (The software can store order data)	Data pemesanan yang terjadi dapat disimpan oleh perangkat lunak (The order data that occurs can be stored by the software)

Contoh Penulisan Kebutuhan Non Fungsional

Contoh 1

Nonfunctional Requirements

1. Operasional

1.1 Sistem harus dapat berjalan di tablet PC yang digunakan oleh tenaga penjualan.

1.2 Sistem harus terhubung dengan sistem manajemen bengkel.

1.3 Sistem harus terhubung ke printer secara nirkabel.

2. Performa

2.1 Sistem harus mendukung 15 tenaga penjualan.

2.2 Sistem harus diperbarui dengan penawaran harga kendaraan setiap 15 menit.

3. Keamanan

3.1 Tidak ada tenaga penjualan yang dapat mengakses kontak pelanggan tenaga penjualan lainnya.

3.2 Hanya pemilik dan manajer penjualan yang dapat menyetujui penawaran pelanggan.

3.3 Penggunaan setiap tablet PC harus dibatasi untuk tenaga penjualan yang ditugaskan.

4. Kebudayaan dan Politik

4.1 Kebijakan perusahaan menyatakan bahwa semua peralatan komputer dibeli dari Dell.

4.2 Informasi pribadi pelanggan dilindungi sesuai dengan Undang-Undang Perlindungan Data.

4.3 Sistem akan sesuai dengan "undang-undang lemon" negara bagian

Contoh 2

Tabel 7 Kebutuhan non Fungsional

ID	Parameter	Kebutuhan
SRS-NF-OUT-023	Availability	Perangkat Lunak harus terus dapat beroperasi 7 hari perminggu, 24 jam per hari tanpa gagal
SRS-NF-PR-024	Reliability	Kegagalan dalam proses transaksi memiliki toleransi satu kali gagal dalam satu minggu
SRS-NF-OUT-025	Ergonomy	Tampilan antarmuka web Super Monster Mall mudah dipahami user dan estetikanya bagus
SRS-NF-PR-026	Portability	Perangkat Lunak dapat dipakai di platofrm Windows dan Linux
SRS-NF-PR-027	Memory	Perangkat Lunak mampu mengirimkan notifikasi e-mail kepada pengguna maksimal dalam waktu 30 detik
SRS-NF-PR-028	Response time	Perangkat Lunak mampu melakukan update data dalam waktu 5 detik
SRS-NF-PR-029	Safety	Perangkat lunak menggunakan standar enkripsi HTTPS

Be S.M.A.R.T.

Specific:Spesifik dan detail

Measurable:Dapat diukur

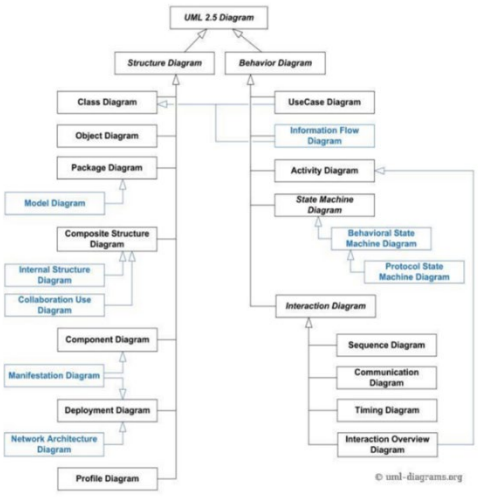
Attainable:Dapat dicapai

Realistic:Realistis

Tangible:Berwujud atau dapat diwujudkan

USE CASE DIAGRAM

Pengertian UML



Unified Modeling LanguageTM (UML®) adalah bahasa pemodelan visual standar (bukan proses pengembangan perangkat lunak) yang digunakan untuk:

- pemodelan bisnis dan sejenis proses
- analisis, desain, dan implementasi sistem berbasis perangkat lunak.

UML menjelaskan proses yang:

a. memberikan panduan untuk urutan kegiatan tim,

b. menentukan apa yang harus dikembangkan artefak,

c. mengarahkan tugas pengembang individu dan tim secara keseluruhan, dan

d. menawarkan kriteria untuk memantau dan mengukur produk dan kegiatan proyek.

Spesifikasi UML

- Diagram struktur menunjukkan struktur statis dari sistem dan bagian-bagiannya pada abstraksi yang berbeda dan tingkat pelaksanaan dan bagaimana mereka berhubungan satu sama lain.
- Diagram perilaku menunjukkan perilaku dinamis dari objek dalam suatu sistem, yang dapat digambarkan sebagai serangkaian perubahan ke sistem dari waktu ke waktu.

UCD (Use Case Diagram)

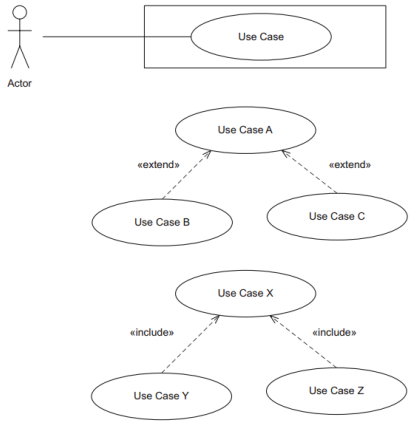


Figure 2.1. UML notation for a use case diagram

Diagram perilaku yang digunakan untuk menggambarkan serangkaian tindakan (Use Case) bahwa beberapa sistem atau system (subjek) dapat melakukan bekerjasama dengan satu atau lebih pengguna eksternal dari sistem (aktor).

Use Case Diagram menceritakan apa yang sistem akan lakukan (what the system will do)

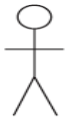
Menekankan "apa" yang diperbuat sistem, dan bukan "bagaimana"

Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem

Menggambarkan kebutuhan sistem dari sudut pandang pengguna (user)

Simbol Use Case

Aktor

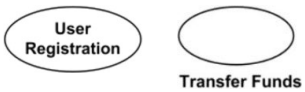


Seorang aktor adalah perilaku yang menentukan peran yang dimainkan oleh entitas eksternal yang berinteraksi dengan subjek (misalnya dengan bertukar sinyal dan data), pengguna manusia dari sistem yang dirancang, beberapa layanan sistem atau perangkat keras menggunakan lain dari subjek. DIgambarkan dengan

Aktor Eksternal

Sebuah pelaku usaha merupakan peran yang dimainkan oleh beberapa **orang atau sistem eksternal** untuk bisnis dimodelkan dan berinteraksi dengan bisnis. Digambarkan mirip seperti "sticky man" dengan garis.

Use Case

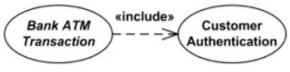


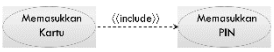
Menggambarkan **fungsionalitas** yang disediakan oleh sistem- sistem yang berasal dari daftar kebutuhan sistem. Digambarkan dengan objek elips. Penamaan use case bisa di dalam atau di bawah elips. Dinyatakan dengan menggunakan **kata kerja** di awal frase nama use case

Business Case

Business use case penggunaan bisnis use case untuk mendukung Pemodelan untuk mewakili fungsi bisnis, proses, atau kegiatan yang dilakukan dalam bisnis model. penggunaan bisnis kasus harus menghasilkan hasil nilai diamati untuk business actor.

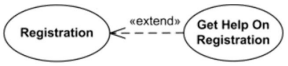
Include



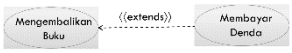


Relasi Include merupakan hubungan berarah antara dua use case yang mana digunakan untuk menunjukkan bahwa tingkah laku dari use case include adalah ditambahkan dalam tingkah laku use case dasar. Relasi antar use case ini merupakan relasi yang diperlukan, tidak opsional. use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai **syarat dijalankan use case ini**

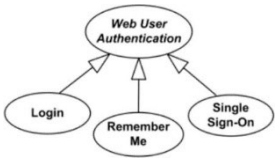
Extend



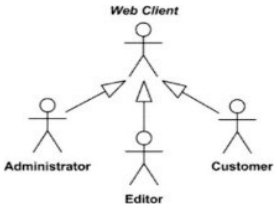
Relasi Extend adalah hubungan berarah yang menentukan bagaimana dan kapan perilaku didefinisikan dalam biasanya **tambahan (opsional)** penggunaan extend use case dapat dimasukkan ke dalam perilaku yang ditetapkan dalam kasus penggunaan yang berkepanjangan. Relasi antar use case ini merupakan relasi yang **optional, sebagai pelengkap**. Use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan



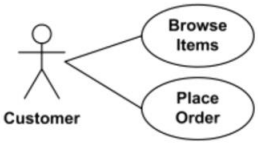
Generalisasi



Relasi Generalisasi merupakan Generalisasi antara use case anak use case **mewarisi** sifat dan perilaku induk use case dan mungkin menimpa induk use case.



Asoasi

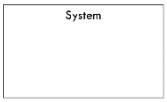


Relasi Asosiasi menghubungkan antara aktor dan use case. Digambarkan dengan garis tanpa anak panah.

Subject

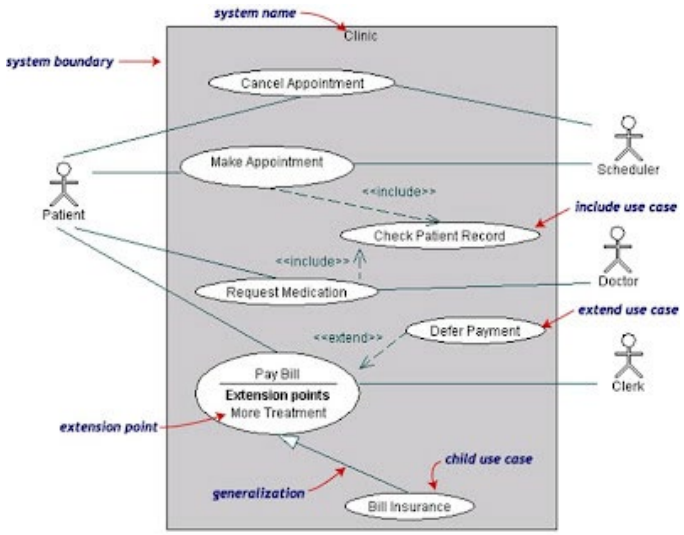
Subjek adalah bisnis, perangkat lunak sistem, subsistem, komponen, perangkat, dll. Hal ini sangat penting untuk menentukan jenis sistem itu, dan apa yang ruang lingkup atau batas.

System



System Boundary dengan menuliskan nama sistem di atas dalam kotak / boundary, menampilkan batasan sistem (scope of the system), actors are outside the scope of the system

Contoh Use Case Diagram



Deskripsi Use Case

Menjelaskan fungsi dasar dari sistem menggunakan kata-kata (words)

- Apa yang dapat dilakukan pengguna
- Bagaimana sistem merespons

Setiap use case harus dijelaskan alur prosesnya melalui sebuah deskripsi use case (use case description) atau scenario use case

Deskripsi use case berisi:

1) Nama use case yaitu penamaan use case yang menggunakan kata kerja
2) Deskripsi yaitu penjelasan mengenai tujuan use case dan nilai yang akan didapatkan oleh aktor
3) Kondisi sebelum (pre-condition) yaitu kondisi-kondisi yang perlu ada sebelum use case dilakukan.
4) Kondisi sesudah (post-condition) yaitu kondisi-kondisi yang sudah dipenuhi ketika use case sudah dilaksanakan
5) Skenario Normal (normal course) yaitu alur yang menceritakan jika semua aksi yang dilakukan adalah benar atau proses yang harusnya terjadi
6) Skenario Alternatif (alternate course) yaitu alur yang menceritakan aksi alternatif, yang berbeda dari skenario normal.

Contoh:

Nama Use-Case:	Make Appointment
Aktor	Patient dan Scheduler
Deskripsi	Membuat temu janji untuk berobat
Normal Course	1. Patient menginisialisasi proses ini dengan membuat temu janji untuk berobat dengna mendatangi tempat praktek 2. Hal ini kemudian dicatat oleh scheduler
Alternate Course	1a. Patient dapat membuat janji lewat telepon 2a. Scheduler dapat menuliskan temu janji berdasarkan informasi yang diberikan oleh patiet.
Pre-Condition:	Patient sakit
Post-Condition:	Cancel Appointment, check patient's record, request medication
Assumption:	-

ACTIVITY DIAGRAM

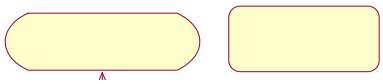
- untuk menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses
- dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis
- Bermanfaat dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan
- Dibuat berdasarkan sebuah atau beberapa use case pada use case diagram
- Berhubungan dengan diagram Statechart. Diagram Statechart fokus pada obyek dalam suatu proses (atau proses menjadi suatu obyek), sedangkan Activity Diagram fokus pada aktifitas-aktifitas yang terjadi yang terkait dalam suatu proses tunggal.
- Menunjukkan bagaimana aktifitas-aktifitas tersebut bergantung satu sama lain
- Menggambarkan aktivitas sistem bukan apa yang dilakukan aktor
- Activity Diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.
- Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Tujuan Activity Diagram

1) Menjelaskan urutan aktivitas dalam suatu proses.
2) Di dalam dunia bisnis biasanya digunakan untuk modeling (memperlihatkan urutan proses bisnis).
3) Mudah dalam memahami proses yang ada dalam sistem secara keseluruhan.
4) Merupakan metode perancangan yang terstruktur, mirip dengan Flowchart maupun Data Flow Diagram (DFD).
5) Mengetahui aktivitas aktor/pengguna berdasarkan use case/diagram yang dibuat sebelumnya.

Simbol Activity Diagram

Activity

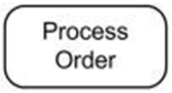


Activity menggambarkan sebuah pekerjaan/tugas dalam workflow.

Pada UML, activity digambarkan dengan simbol belah ketupat='lozenge' (horizontal top and bottom with convex sides).

Activity State

Action



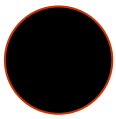
Action/tindakan digambarkan sebagai persegi panjang dengan ujung melingkar. Nama tindakan atau deskripsi lainnya mungkin muncul dalam symbol.

State Transitions

State transition menunjukkan kegiatan apa berikutnya setelah suatu kegiatan sebelumnya

Pada UML, state transition digambarkan oleh sebuah *solid line* dengan panah

Start State



Start state dengan tegas menunjukkan dimulainya suatu workflow pada sebuah activity diagram

Pada UML, start state digambarkan dengan simbol lingkaran yang solid

Activity Initial Node

Initial node/Simpul awal adalah node kontrol dimana arus dimulai saat aktivitas dipanggil. Aktivitas mungkin memiliki lebih dari satu simpul awal.

Simpul awal ditampilkan sebagai lingkaran padat kecil.

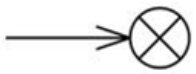
End State

End state menggambarkan akhir atau terminal dari pada sebuah activity diagram

Bisa terdapat lebih dari satu end state pada sebuah activity diagram

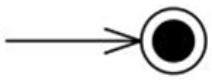
Pada UML, end state digambarkan dengan simbol sebuah bull's eye (mata sapi)

Flow Final Node



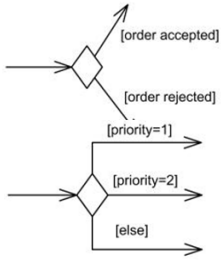
Flow final node/Arus simpul akhir adalah node akhir kontrol yang mengakhiri aliran. Notasi untuk node arus akhir adalah lingkaran kecil dengan X di dalamnya.

Activity Final Node



Activity final/Aktivitas simpul terakhir adalah node akhir control yang menghentikan semua arus dalam suatu kegiatan. Aktivitas simpul terakhir ditampilkan sebagai lingkaran padat dengan lingkaran berongga di dalamnya.

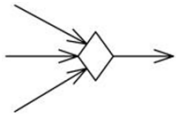
Decision



Decision/Keputusan digambarkan oleh node dengan dua tepi keluar. suatu titik/point pada activity diagram yang mengindikasikan suatu kondisi dimana ada kemungkinan perbedaan transisi Pada UML, decision digambarkan dengan sebuah simbol diamond Decision node adalah node kontrol yang menerima token pada satu atau dua sisi yang masuk dan memilih satu tepi keluar dari satu atau lebih arus keluar. Keputusan node dengan tiga tepi keluar dan [else]. Untuk poin keputusan, panah "ELSE" yang telah ditentukan dapat didefinisikan paling banyak satu tepi keluar.

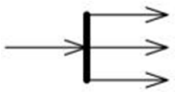
Pada penggunaan percabangan (decision) dengan fork. decision digunakan untuk memecah aktivitas yang bersifat kondisional. Contohnya pilihan **Ya** atau **Tidak**, jika opsi **Ya**, maka terjadi aksi baru dan jika **Tidak**, maka menolak aksi baru.

Merge



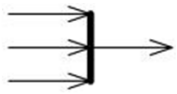
Merge/Gabungan simpul dengan tiga sisi yang masuk dan tepi keluar tunggal. Merge node adalah node kontrol yang menyatukan beberapa arus masuk untuk menerima arus keluar tunggal. Tidak ada penggabungan token. Gabung seharusnya tidak digunakan untuk menyinkronkan arus bersamaan. **Menggabungkan flow yang sudah dipecah menjadi beberapa bagian oleh suatu flow**

Fork



Simpul fork/garpu dengan tepi aktivitas tunggal yang memasukinya, dan tiga sisi meninggalkannya. Fork node adalah node kontrol yang memiliki satu edge yang masuk dan beberapa tepi keluar dan digunakan untuk membagi arus masuk menjadi beberapa aliran bersamaan. Notasi untuk simpul garpu adalah segmen garis dengan tepi aktivitas tunggal yang memasukinya, dan dua atau lebih ujungnya meninggalkannya. **Menunjukkan kegiatan yang dilakukan secara parallel**, memecah behaviour menjadi aktivitas yang paralel, contohnya seperti pengguna dapat memilih, menambah, mengubah, serta bisa juga menghapus.

Join



Join node/Node gabungan dengan tiga sisi aktivitas yang masuk dan satu sisi meninggalkannya. **Untuk menunjukkan kegiatan yang digabungkan.**

Penggabungan simpul adalah node kendali yang memiliki beberapa tepi masuk dan satu tepi keluar dan digunakan untuk menyinkronkan arus masuk bersamaan. Notasi untuk node join adalah segmen garis dengan beberapa tepi aktivitas yang memasukinya, dan hanya satu sisi yang meninggalkannya

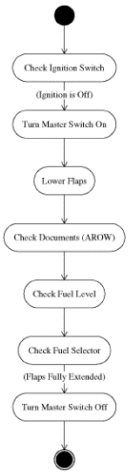
Swimlanes

Object swimlane untuk menggambarkan **objek** mana yang **bertanggung jawab untuk aktivitas tertentu.**

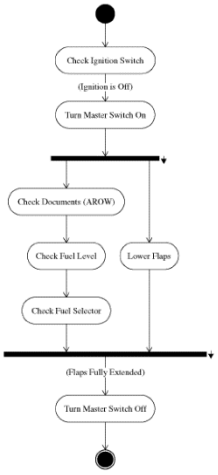
Tahap membuat *Activity Diagram*

1) Mulailah dengan node awal untuk start state atau titik awal.
2) Tambahkan partisi jika itu memang relevan untuk analisis yang akan dibuat.
3) Buatlah suatu aksi untuk setiap langkah utama dari use case.
4) Tambahkan alur (flow) dari setiap aksi ke aksi lainnya. Keputusan berada di node akhir. Setiap aksi hanya mendapat satu alur masuk dan satu alur keluar yang nantinya menuju ke forks, joins, decisions, dan merges.
5) Tambahkan juga percabangan atau decision bila alur dipecah menjadi suatu kondisi pilihan. Jangan lupa untuk menggabungkannya kembali dengan merge.
6) Menambahkan forks dan joins jika aktivitas dilakukan secara paralel.
7) Langkah yang terakhir yaitu akhiri proses dengan notasi akhir atau end state.

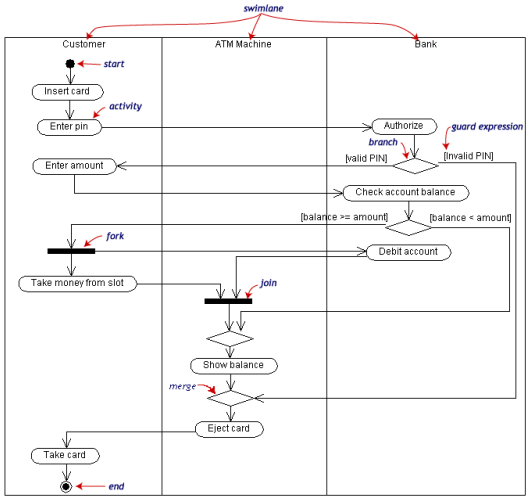
Contoh tanpa percabangan



Contoh dengan Percabangan



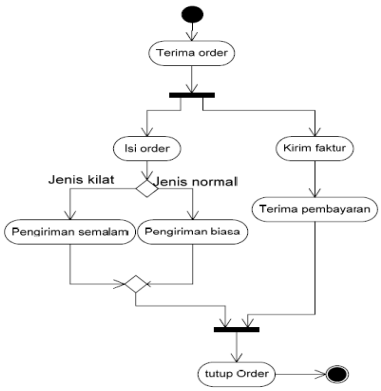
Contoh kasus "Pengambilan uang dari bank melalui ATM"



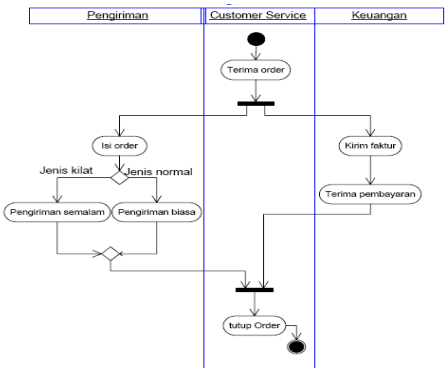
Ada tiga aktifitas kelas (orang, dan lainnya) yang terkait, yaitu : Customer, ATM, and Bank.

Proses berawal dari lingkaran start hitam pada bagian atas dan berakhir di pusat lingkaran stop hitam/putih pada bagian bawah. Aktivitas digambarkan dalam bentuk kotak persegi.

Contoh Kasus Sistem Order



Dengan *Swimlane*



UTS

1. Jelaskan mengapa dibutuhkan dokumen Software Requirement Specifications dan sebutkan meliputi apa saja dokumen Software Requirement Specifications!

Jawab:

Dokumen Software Requirement Specifications (SRS) sangat dibutuhkan dalam pengembangan perangkat lunak karena SRS berfungsi sebagai dasar kesepakatan antara klien dan developer tentang apa yang akan dibangun. SRS menjadi acuan yang jelas dan terstruktur untuk memastikan semua pihak memiliki pemahaman yang sama tentang produk akhir.

Berikut beberapa alasan mengapa SRS sangat dibutuhkan:

- **Meminimalisir kesalahan interpretasi:** SRS mendefinisikan kebutuhan software secara detail dan terstruktur, sehingga mengurangi potensi kesalahpahaman antara klien dan developer.
- **Memudahkan estimasi biaya dan waktu:** Dengan SRS yang jelas, developer dapat mengestimasi biaya dan waktu pengembangan secara lebih akurat.
- **Memudahkan proses validasi dan verifikasi:** SRS menjadi acuan untuk menguji dan memvalidasi apakah software yang dibangun sudah sesuai dengan kebutuhan yang telah disepakati.
- **Memudahkan transfer pengetahuan:** SRS dapat digunakan untuk mentransfer pengetahuan tentang software kepada pihak lain, misalnya jika terjadi pergantian personil developer.
- **Mendasari pengembangan dan perbaikan software:** SRS menjadi acuan untuk melakukan pengembangan dan perbaikan software di kemudian hari.

Dokumen SRS umumnya meliputi hal-hal berikut:

- **Pendahuluan**: Berisi latar belakang, tujuan, ruang lingkup, batasan masalah, nama software, definisi dan singkatan, referensi, dan penjelasan umum (uraian singkat dan fitur software).
- **Gambaran Umum**: Meliputi karakteristik pengguna, jenis pengguna, hak akses pengguna, dan ketergantungan software.
- **Analisis Kebutuhan**: Berisi identifikasi aktor, identifikasi use case, diagram use case, skenario use case, dan rencana antarmuka sistem.
- **Non-Functional Requirements**: Berisi kebutuhan non-fungsional seperti performa, keamanan, kegunaan, dll.
- **Kontrak Kerja**: Berisi informasi tentang biaya software, kontrak dan perjanjian antara klien dan developer.

Dengan mencantumkan semua informasi penting ini dalam dokumen SRS, proses pengembangan software dapat berjalan lebih terstruktur, efektif, dan meminimalisir potensi konflik di kemudian hari.

2. Jelaskan mengapa Use Case Diagram dan Activity Diagram termasuk dalam Behavior Diagram!

Jawab:

Use Case Diagram dan Activity Diagram termasuk dalam Behavior Diagram karena keduanya sama-sama menggambarkan perilaku sistem. Namun, mereka fokus pada aspek yang berbeda:

Use Case Diagram:

- Berfokus pada **interaksi antara sistem dan aktor (pengguna atau sistem eksternal)**.
- Menjelaskan **fungsionalitas sistem dari sudut pandang pengguna**.
- Menunjukkan **apa** yang dilakukan sistem, bukan **bagaimana** cara melakukannya.

Activity Diagram:

Berfokus pada alur kerja dan urutan aktivitas dalam suatu proses.

- Menjelaskan **bagaimana** suatu tugas diselesaikan, termasuk **keputusan, percabangan, dan aliran paralel.**
- Dapat menggambarkan proses bisnis atau alur kerja internal sistem.

Secara sederhana, Use Case Diagram menunjukkan "apa yang dilakukan sistem", sedangkan Activity Diagram menunjukkan "bagaimana sistem melakukannya".

Keduanya merupakan bagian penting dalam pemodelan perilaku sistem karena memberikan perspektif yang berbeda dan saling melengkapi tentang cara kerja sistem.

3. Jika anda akan membuat rancangan sistem, sebagai sistem analis yang Anda lakukan adalah: (pilih salah satu (a atau b) dan jelaskan pilihan Anda!)
   a. Menganalisis proses bisnis (FR dan NFR), Use Case Diagram, Activity Diagram
   b. Menganalisis proses bisnis (FR dan NFR), Activity Diagram, Use Case Diagram

Jawaban yang lebih tepat adalah **a. Menganalisis proses bisnis (FR dan NFR), Use Case Diagram, Activity Diagram.**

Berikut penjelasannya:

Menganalisis proses bisnis (FR dan NFR): Tahap awal yang krusial adalah memahami proses bisnis yang ingin dimodelkan. Ini termasuk mengidentifikasi kebutuhan fungsional (FR) yang menjelaskan apa yang harus dilakukan sistem, dan kebutuhan non-fungsional (NFR) yang menjelaskan batasan dan kualitas sistem.

Use Case Diagram: Setelah memahami proses bisnis, Use Case Diagram dibuat untuk memvisualisasikan interaksi antara aktor (pengguna atau sistem eksternal) dengan sistem. Diagram ini menunjukkan fungsionalitas sistem dari sudut pandang pengguna, menggambarkan "apa" yang dilakukan sistem, bukan "bagaimana".

Activity Diagram: Tahap terakhir adalah membuat Activity Diagram untuk menggambarkan alur kerja dan urutan aktivitas dalam setiap use case yang telah diidentifikasi. Diagram ini menunjukkan "bagaimana" sistem menjalankan fungsi-fungsi yang dijelaskan dalam Use Case Diagram.

Mengapa urutan ini lebih tepat?

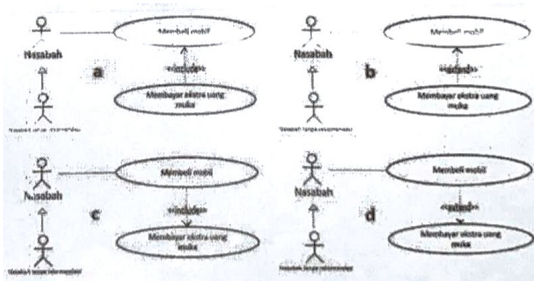
Membuat Use Case Diagram sebelum Activity Diagram memastikan bahwa:

- Fokus awal diarahkan pada fungsionalitas sistem dari sudut pandang pengguna.
- Use Case Diagram menjadi dasar untuk mengembangkan Activity Diagram yang lebih detail.
- Setiap Activity Diagram memiliki konteks yang jelas, yaitu menggambarkan alur kerja dari use case tertentu.

Dengan demikian, alur kerja sistem analis menjadi lebih terstruktur dan sistematis, dimulai dari pemahaman kebutuhan, visualisasi interaksi pengguna, hingga detail alur kerja internal sistem.

4. Terdapat dua pernyataan: Nasabah Bank X dapat melakukan pembelian mobil. Nasabah yang tidak mendapat rekomendasi kredit harus membayar ekstra uang muka. Pada gambar di bawah ini terdapat 4 rancangan (a, b, c, d)

Pilihlah satu rancangan diagram Use Case di bawah dan sertakan penjelasannya.



a

nasabah yang merekomendasi----->Nasabah-------Membeli mobil<---<<include>>---Membayar adanya uang muka

b

nasabah yang merekomendasi----->Nasabah-------Membeli mobil<---<<extend>>---Membayar adanya uang muka

c

nasabah yang merekomendasi----->Nasabah-------Membeli mobil---<<include>>--->Membayar adanya uang muka

d

nasabah yang merekomendasi----->Nasabah-------Membeli mobil---<<extend>>--->Membayar adanya uang muka

Jawab:

Pada kasus ini, "Membayar adanya uang muka" dianggap sebagai tindakan yang opsional atau tidak wajib dilakukan saat "Membeli mobil".

Relasi <<extend>> pada diagram use case menunjukkan bahwa use case "Membayar adanya uang muka" merupakan ekstensi atau perpanjangan yang bersifat opsional dari use case "Membeli mobil".

Artinya, jika nasabah mendapat rekomendasi kredit, maka nasabah dapat langsung "Membeli mobil" tanpa harus "Membayar adanya uang muka" terlebih dahulu.

Sedangkan jika nasabah tidak mendapat rekomendasi kredit, barulah use case "Membayar adanya uang muka" akan dijalankan sebagai syarat tambahan sebelum "Membeli mobil".

Jadi pembayaran uang muka bersifat opsional/kondisional, tergantung apakah nasabah mendapat rekomendasi kredit atau tidak saat akan membeli mobil.

5. Seorang sistem analis merekam hasil diskusi meeting antara tim pengembang Perangkat Lunak dengan klien mereka sebuah divisi pengarsipan di perusahaan X sebagai berikut:

   “Sistem pengarsipan harus tersedia sepanjang waktu (24 jam/7 hari). Siapapun dapat menelusuri katalog arsip, tetapi bila ingin unduh/unggah dokumen, identitasnya harus tercapat sistem. Identitas ditunjukkan dengan alamat email dan kata kunci sepanjang 8 karakter. Apabila identitas tidak cocok dengan data anggota terdaftar, mereka akan diminta untuk mendaftar dahulu. Seorang staf divisi akan ditunjuk sebagai pengelola sistem, yang dapat membatalkan keanggotaan seseorang, namun seorang anggota tidak dapat membatalkan keanggotaannya sendiri. Setiap akhir minggu, sebuah laporan aktivitas sistem harus dihasilkan untuk diketahui kepala bagian divisi. Desain antarmuka yang dibuat harus mematuhi regulasi Akses Disabilitas.”

   Dari penjelasan di atas, desain sebuah Diagram Use Case sesuai dengan **functionality requirements** yang tercatat oleh sang sistem analis. Kemudian, pisahkan dan tulis requirements lain berupa **non-functionality requirements** di bawah Diagram Use Case Anda.

   Jawab:

Functional Requirements (FR)

ID	Kebutuhan	Penjelasan
FR-01	Sistem harus dapat diakses untuk penelusuran katalog arsip oleh siapapun.	Semua pengguna, baik terdaftar maupun tidak, dapat melihat katalog arsip.
FR-02	Sistem harus memungkinkan pengguna terdaftar untuk mengunduh dokumen.	Hanya pengguna dengan akun terdaftar yang dapat mengunduh dokumen.
FR-03	Sistem harus memungkinkan pengguna terdaftar untuk mengunggah dokumen.	Hanya pengguna dengan akun terdaftar yang dapat mengunggah dokumen.
FR-04	Sistem harus menyediakan fitur pendaftaran pengguna.	Pengguna dapat mendaftar untuk mendapatkan akun dengan email dan kata kunci.
FR-05	Sistem harus memvalidasi identitas pengguna saat mengunduh/unggah dokumen.	Sistem akan memeriksa kecocokan email dan kata kunci dengan data anggota terdaftar.
FR-06	Sistem harus mengarahkan pengguna yang tidak terdaftar untuk mendaftar saat mencoba mengunduh/unggah dokumen.	Pengguna tanpa akun akan diminta mendaftar terlebih dahulu.
FR-07	Sistem harus memungkinkan staf divisi (pengelola sistem) untuk membatalkan keanggotaan pengguna.	Staf divisi memiliki hak untuk membatalkan akun pengguna.
FR-08	Sistem tidak mengizinkan anggota untuk membatalkan keanggotaannya sendiri.	Anggota tidak memiliki hak untuk membatalkan akunnya sendiri.
FR-09	Sistem harus menghasilkan laporan aktivitas sistem setiap akhir minggu.	Laporan aktivitas sistem akan tersedia untuk kepala bagian divisi setiap minggu.