



# Analisis Perancangan Sistem Informasi

Tahun 2025

## Modul 1

Functional Requirement, Use Case Diagram & Activity Diagram

Arranged By:  
**EAD Laboratory Team**

# Asisten Praktikum



ENTERPRISE  
APPLICATION  
DEVELOPMENT



**ARFY**  
Rifah Arfiyah



**JHON**  
I Gede Made Ari A.



**CIAA**  
Allicia Felicitas G.



**PALS**  
Naufal Akmal R.



**ICEA**  
Naisya Najmi



**RAYA**  
Muhammad Raya R.



**ASWW**  
Aswanga P.



**FRHN**  
Muhammad Ilyas



**ALEM**  
Matthew Alexander



**BGND**  
Stefanus Jesano P.



**YARE**  
Muhammad Ikhtiar



**NAVY**  
Neisy Nayyara A.



**AHRI**  
Balqis Eka N.



**MAUL**  
Maulida Afifi U.



**RARA**  
Annisa Agustina



**UCIK**  
Suci Larasati



**ZEAL**  
Azel Pandya M.



**LYAA**  
Yesitra Anugrah A.



**RRAS**  
Raras Nurhaliza H.



**NASA**  
Niken Ayu S.



**AARN**  
Aaron James E.



**ARCH**  
Ario Rafa M.

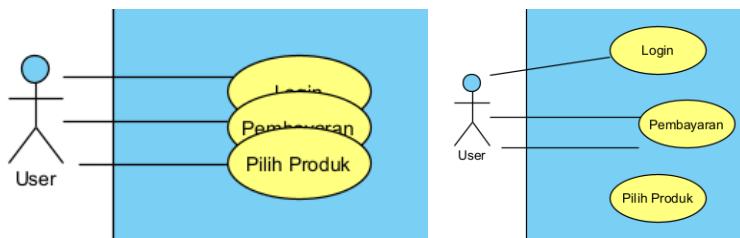
## DAFTAR ISI

<b>DAFTAR ISI</b>	<b>3</b>
<b>PERATURAN PRAKTIKUM</b>	<b>4</b>
<b>LANDASAN TEORI</b>	<b>7</b>
A. Functional Requirement	7
B. Use Case Diagram	7
1. Definisi dan Manfaat	7
2. Karakteristik Use Case Diagram	7
3. Notasi Use Case Diagram	8
4. Use Case Description	11
C. Activity Diagram	13
1. Definisi dan Manfaat	13
2. Karakteristik Activity Diagram	13
3. Notasi Activity Diagram	13
<b>LANGKAH-LANGKAH PRAKTIKUM</b>	<b>18</b>
A. Cara Mendokumentasikan Functional Requirement	18
B. Cara Membuat Use Case Diagram	18
C. Cara Membuat Activity Diagram	26
<b>CONTOH STUDI KASUS</b>	<b>40</b>
<b>DAFTAR PUSTAKA</b>	<b>41</b>

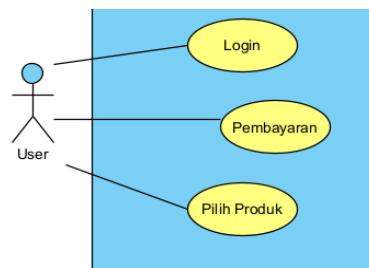
## PERATURAN PRAKTIKUM

1. Setiap peserta praktikum harus datang tepat waktu sesuai dengan jadwal. Toleransi keterlambatan hadir 15 menit. Jika melebihi batas waktu tersebut diperkenankan mengikuti praktikum, namun tidak mendapatkan tambahan waktu dan/atau nilai.
2. Perizinan Praktikum :
  - a. Praktikan yang ingin melakukan perizinan diwajibkan untuk menghubungi asisten praktikum group plotingan, kemudian menghubungi komisi disiplin dan mengisi form perizinan.
  - b. Form perizinan dapat diakses melalui LMS atau dikirimkan oleh komisi disiplin.
  - c. Izin berkaitan dengan Sakit atau Kemalangan , maka praktikan dapat memberikan surat perizinan kepada pihak Komisi Disiplin maksimal 3 hari setelah jadwal (shift) praktikum.
  - d. Izin berkaitan dengan Kematian dapat memberikan surat yang ditandatangani oleh keluarga terdekat.
  - e. Izin lomba atau penugasan institusi tidak berlaku apabila tidak terdapat bukti dispensasi dari Igracias. NB : screenshot dispensasi dari igracias wajib dilampirkan dan dikirim melalui form perizinan yang ada di LMS atau didapat dari komisi disiplin.
3. Seragam Praktikum :
  - a. Mahasiswa wajib menggunakan celana bahan hitam (bukan chino atau jeans) pada saat praktikum.
  - b. Mahasiswi wajib menggunakan rok hitam/biru gelap panjang tidak ketat pada saat praktikum
  - c. Dresscode praktikum (Mengikuti Peraturan Telkom)
    - Senin : Mengenakan kemeja merah telkom atau kemeja putih polos
    - Selasa s/d Rabu : Mengenakan kemeja putih Telkom atau kemeja putih polos
    - Kamis s/d Sabtu : Mengenakan kemeja formal berkerah (bukan kerah sanghai dan bukan polo)
    - Jumat : Mengenakan kemeja/ baju batik bukan outer.
  - d. Jika terdapat kendala dalam baju seragam maka praktikan diperbolehkan mengganti menggunakan kemeja putih Telkom atau kemeja putih polos.
  - e. Praktikan dilarang untuk menggunakan aksesoris berlebih, seperti kalung, gelang, piercing, dan lain-lain. Kecuali aksesoris keagamaan.
  - f. Membuka sepatu saat memasuki ruangan lab.
  - g. Untuk pengecekan seragam bagi praktikan yang masih menggunakan topi atau jaket harap dilepas terlebih dahulu.
4. Peraturan Pengerjaan
  - a. Post test dilakukan secara individu dan dilarang membuka modul.
  - b. Jika praktikan ketahuan membuka modul, searching di internet, menggunakan AI, dan alat komunikasi lainnya, nilai post test akan menjadi 0.

- c. Studi Kasus dikerjakan secara individu.
- d. Jawaban tidak boleh sama dengan setiap individu.
- e. Setiap diagram pada studi kasus wajib dibuat dengan memperhatikan kerapian agar mudah dipahami oleh asisten praktikum. Contoh diagram yang dianggap kurang rapi seperti notasi diagram yang bertumpuk-tumpuk, arah asosiasi yang tidak jelas, dan lain sebagainya.  
(Penggambaran diagram yang kurang rapi)



(Penggambaran diagram yang rapi)



- f. Penggambaran diagram yang kurang rapi akan mendapat pengurangan nilai maksimal 7 poin dan penggambaran diagram yang rapi akan mendapat penambahan nilai maksimal sebesar 5 poin.
- g. Pengerjaan jurnal dilarang searching di internet, menggunakan AI, bertanya kepada asisten praktikum dan berdiskusi dengan praktikan lainnya, dan alat komunikasi, melakukan hal tersebut akan dikenakan plagiarisme.
- h. Submit hasil pengerjaan berupa file format PDF
- i. Format Pengumpulan Hasil Pengerjaan:
  - i. File penamaan Jurnal dalam bentuk PDF:  
APSI\_MODULX\_KODEASISTEN\_NAMA LENGKAP\_NIM  
Contoh:  
APSI\_MODUL1\_JHON\_ARI ANANTA\_1021202200000
  - ii. File penamaan Tugas Pendahuluan dalam bentuk PDF:  
APSI\_TPX\_KODEASISTEN\_NAMA LENGKAP\_NIM  
Contoh:  
APSI\_TP1\_JHON\_ARI ANANTA\_1021202200000
- j. Screenshot hasil diagram tiap nomor dalam bentuk fullscreen dan menampilkan tanggal serta waktu. Jika tidak fullscreen maka akan dipotong 10% per nomor.

- k. Salah format nama pada file pengerojan nilai modul akan dipotong sebesar 10%.
- l. Jika salah mengumpulkan file nilai akan dipotong sebesar sebesar 50%.
- m. Terlambat mengumpulkan file pengerojan Jurnal dibawah 2 menit saat jurnal, maka nilai jurnal akan dipotong sebesar 0% . (Jika terjadi gangguan bersama)
- n. Terlambat mengumpulkan file pengerojan Jurnal nilai jurnal akan dipotong sebesar 15% .
- o. Terlambat mengumpulkan file pengerojan Tugas Pendahuluan nilai Tugas Pendahuluan akan dipotong sebesar 35%.
- p. Tidak mengumpulkan Tugas Pendahuluan maka keseluruhan modul tersebut akan dipotong 70% .
- q. Segala alat komunikasi harap dimatikan dan dimasukkan kedalam tas, dan tas disimpan dibawah meja.
- r. Jika ada perangkat praktikum yang bermasalah dapat menghubungi asprak yang bertugas.
- s. Segala bentuk kecurangan dan plagiarisme akan diproses ke komisi disiplin dan nilai akhir modul menjadi 0.

## LANDASAN TEORI

### A. Functional Requirement

Functional Requirement adalah pernyataan tentang apa yang harus dilakukan atau karakteristik apa yang harus dimiliki oleh sistem. Persyaratan sistem yang lebih teknis dari sudut pandang user yang disebut dengan Functional Requirement.

### B. Use Case Diagram

#### 1. Definisi dan Manfaat

Use case diagram adalah salah satu jenis diagram dalam Unified Modeling Language (UML) yang digunakan untuk menggambarkan interaksi antara pengguna (aktor) dan sistem. Diagram ini memberikan gambaran tentang fungsi-fungsi yang dapat dilakukan oleh sistem dan bagaimana pengguna berinteraksi dengan fungsi-fungsi tersebut. Use case diagram tidak menjelaskan secara rinci bagaimana sistem beroperasi, tetapi lebih menekankan pada "apa" yang dilakukan oleh sistem dari perspektif pengguna.

Beberapa manfaat dari use case diagram diantaranya adalah:

1. Mengidentifikasi aktor yang dapat berinteraksi dengan sistem dan peran masing-masing.
2. Menggambarkan hasil dari functional requirement.
3. Memfasilitasi komunikasi antara analis, pengembang, dan pemangku kepentingan dengan representasi visual.
4. Menyediakan pandangan eksternal yang membantu memahami bagaimana sistem berfungsi dari perspektif pengguna luar.

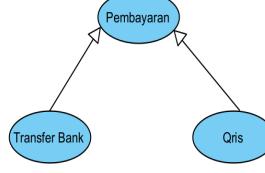
#### 2. Karakteristik Use Case Diagram

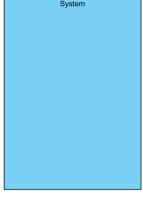
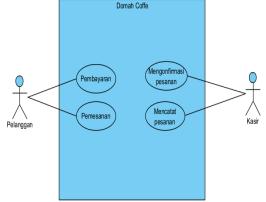
Use case diagram memiliki sejumlah karakteristik utama yang menjadikannya penting dalam pemodelan sistem, yaitu :

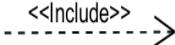
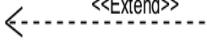
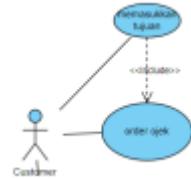
1. Setiap use case dipicu oleh aktor, yang dapat berupa individu, organisasi, atau perangkat lain yang berinteraksi dengan sistem.
2. Use case harus memberikan hasil bernilai atau manfaat minimal kepada aktor.
3. Use case diagram menggambarkan dialog atau interaksi antara aktor (pengguna atau sistem eksternal) dengan sistem.

### 3. Notasi Use Case Diagram

Nama	Notasi	Contoh	Keterangan
Use Case			<p>Mewakili peran manusia, sistem lain, atau perangkat yang berinteraksi dengan <i>use case</i>.</p> <p>Contoh terdapat sebuah <i>use case</i> dengan nama "Pemesanan" yang ditampilkan dalam bentuk oval. <i>Use case</i> ini menggambarkan satu aktivitas utama dalam sistem, yaitu melakukan pemesanan barang atau jasa.</p>
Actor			<p>Menggambarkan aktivitas yang dilakukan aktor.</p> <p>Contoh Terlihat adanya sebuah aktor bernama "Pelanggan" yang terhubung dengan beberapa <i>use case</i>, yaitu "Pemesanan", "Pembayaran", dan "Memberi Ulasan". Hal ini menunjukkan bahwa satu aktor dapat melakukan berbagai aktivitas dalam sistem.</p>
Association			Menghubungkan aktor dengan <i>use case</i> .

			<p>Contoh Terdapat hubungan antara aktor "Pelanggan" dengan use case "Pembayaran" yang ditunjukkan melalui kardinalitas one-to-many. Hal ini berarti bahwa satu pelanggan dapat melakukan lebih dari satu kali pembayaran. Kardinalitas ditampilkan melalui simbol "1" pada aktor dan "*" pada use case, yang menggambarkan bahwa dalam sistem tersebut, seorang pelanggan dapat melakukan banyak transaksi atau pembayaran.</p>
Generalization	→		<p>Menunjukkan bahwa aktor yang lebih spesifik dapat mewarisi peran dari aktor yang lebih umum.</p> <p>Contoh terdapat use case "Pembayaran" yang memiliki dua sub-aktivitas, yaitu "Transfer Bank" dan "QRIS". Hubungan antara use case utama dan sub-aktivitas ditunjukkan dengan panah generalization yang menggambarkan bahwa "Transfer</p>

			<p>"Bank" dan "QRIS" merupakan bentuk spesialisasi dari proses pembayaran. Hal ini berarti, pengguna dapat memilih salah satu dari metode pembayaran tersebut ketika melakukan transaksi.</p>
System	 		<p>Membatasi <i>use case</i> dari interaksi eksternal sistem.</p> <p>Contoh terdapat sebuah <i>system boundary</i> yang menggambarkan ruang lingkup sistem. Didalamnya terdapat beberapa <i>use case</i> seperti "Melihat Menu", "Memesan", "Membayar", dan "Keluar". Aktor "Pelanggan" dapat melakukan semua aktivitas tersebut, sementara aktor "Kasir" hanya berhubungan dengan <i>use case</i> "Membayar".</p>

Include			<p>Menandakan bahwa suatu use case merupakan bagian atau fungsionalitas dari use case lain.</p> <p>Contoh terdapat dua use case yaitu "Beli Kopi" dan "Login", dengan relasi &lt;&lt;include&gt;&gt;. Hal ini menunjukkan bahwa dalam proses Beli Kopi, wajib dilakukan aktivitas Login agar pembelian dapat diproses atau diselesaikan.</p>
Extend			<p>Menunjukkan bahwa suatu use case menyediakan fungsionalitas tambahan dari use case lain jika kondisi tertentu terpenuhi.</p> <p>Pada contoh gambar, terdapat use case "Pay Ojek" yang memiliki relasi &lt;&lt;extend&gt;&gt; dengan use case "Give a Tip". Hal ini berarti fungsionalitas Give a Tip merupakan tambahan yang</p>

			akan dijalankan jika syarat atau kondisi tertentu pada proses Input Menu terpenuhi.
--	--	--	---

#### 4. Use Case Description

Use Case Description adalah dokumen yang digunakan untuk menggambarkan interaksi antara pengguna (aktor) dan sistem dalam situasi tertentu yang disebut skenario. Dokumen ini penting dalam pengembangan perangkat lunak karena menjelaskan bagaimana suatu fitur bekerja secara rinci. Biasanya, Use Case Description mencakup elemen-elemen penting seperti nama use case, deskripsi singkat skenario, aktor yang terlibat, kondisi awal sebelum skenario dimulai, kondisi akhir setelah skenario selesai, serta kemungkinan kondisi error yang dapat terjadi. Selain itu, dokumen ini juga mencantumkan pemicu kejadian, alur standar sebagai langkah-langkah utama, dan alur alternatif jika terjadi penyimpangan dari skenario utama. Dengan adanya Use Case Description, tim pengembang dapat memahami bagaimana sistem beroperasi secara menyeluruh dan mengantisipasi berbagai kondisi yang mungkin terjadi.

Beberapa manfaat *use case description* adalah :

1. Menggambarkan secara rinci bagaimana pengguna berinteraksi dengan sistem.
2. Memberikan deskripsi yang sistematis mengenai alur proses dalam sistem.
3. Menjadi acuan dalam membuat skenario uji coba sistem.
4. Membantu menjembatani komunikasi antara analis, pengembang, penguji, dan pemangku kepentingan lainnya dengan bahasa yang mudah dimengerti.

Cara pengisian *use case description* di antara lain :

Name	Menyebutkan nama dari use case.
Description	Menyediakan gambaran umum tentang apa yang dilakukan oleh use case ini.
Precondition	Menjelaskan kondisi atau status sistem yang harus ada sebelum use case ini dapat dijalankan.
Postcondition	Merupakan keadaan yang diharapkan setelah use case selesai dijalankan.
Error Situation	Menggambarkan kondisi atau situasi ketika terjadi kesalahan selama pelaksanaan use case.
System state in the event of an error	Menggambarkan bagaimana sistem harus berperilaku atau keadaan sistem setelah terjadinya kesalahan.
Actor	Menunjukkan siapa yang terlibat dalam use case ini.
Trigger	Menjelaskan apa yang menyebabkan use case ini dimulai
Standard in Process	Merupakan urutan langkah-langkah yang diikuti selama eksekusi use case yang berhasil
Alternative Processes	Menjelaskan jalur alternatif atau penyimpangan dari proses standar.

## C. Activity Diagram

### 1. Definisi dan Manfaat

*Activity diagram* adalah diagram yang digunakan untuk memodelkan berbagai proses dalam suatu sistem. Diagram ini menggambarkan urutan proses secara vertikal, menunjukkan bagaimana aktivitas dalam sistem berjalan. *Activity diagram* dikembangkan dari *Use Case Diagram* dan memiliki alur aktivitas yang jelas.

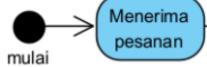
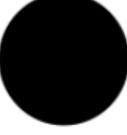
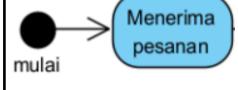
Diagram use case membantu dalam mengidentifikasi aktor-aktor yang terlibat dalam sistem dan skenario penggunaan yang berbeda, membantu memahami siapa yang terlibat dalam proses bisnis dan apa yang mereka coba capai. Sementara itu *Activity diagram* dapat digunakan untuk mendokumentasikan atau menggambarkan alur aktivitas-aktivitas setiap use case yang terdapat di use case diagram.

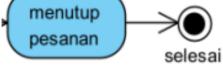
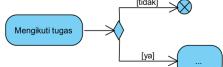
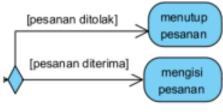
## 2. Karakteristik Activity Diagram

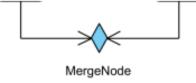
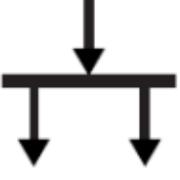
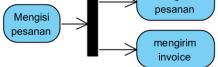
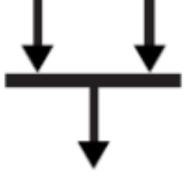
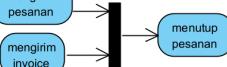
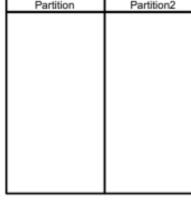
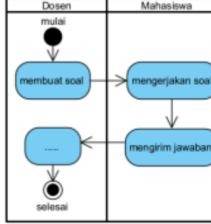
- Memodelkan Alur Kerja Sistem
- Menampilkan aliran kontrol (control flow) dan aliran data (object flow)
- Mendukung Pengambilan Keputusan dan Percabangan
- Memodelkan Aktivitas Paralel dan Sinkronisasi
- Menggunakan Notasi Standar UML

## 3. Notasi Activity Diagram

Nama	Notasi	Deskripsi	Contoh	Keterangan
Action		Perilaku sederhana yang tidak dapat dibagi-bagi lagi.		Menerima pesanan merupakan sebuah tindakan sederhana yang mana informasi pesanan diterima.
Activity		Serangkaian aksi yang membentuk suatu tindakan atau proses.		Terdapat contoh aktivitas proses pemesanan di samping, yang melibatkan berbagai tindakan yang mewakili seluruh proses pemesanan.
Object Node		Representasi objek atau barang dalam sebuah proses.		Contoh invoice di samping adalah salah satu objek yang diberikan/terbentuk dalam proses, menunjukkan

				pencatatan transaksi yang berikan.
Control Flow	→	Urutan eksekusi yang menunjukkan bagaimana proses dilakukan.		Pada gambar di samping terdapat urutan/alur control flow yang dimulai dengan tahapan Mulai, kemudian dilanjutkan dengan tindakan Menerima Pesan.
Object Flow	→	Menunjukkan bagaimana objek bergerak atau dialirkan dari satu aktivitas ke aktivitas lain dalam proses.		Ketika mengalirkan dari tindakan Mengirim Invoice menuju objek Invoice, notasi yang digunakan adalah Object Flow yang menunjukkan perpindahan objek dari satu aktivitas ke aktivitas lainnya.
Initial Node		Awal dari sebuah aktivitas atau proses yang menandakan dimulainya suatu proses.		Initial mode Mulai pada gambar di samping menunjukkan titik awal di mana eksekusi proses dimulai, dilanjutkan dengan tindakan

				menerima pesan.
Final-activity Node		Penanda akhir dari suatu aktivitas atau proses.		Ketika seluruh proses sudah berakhir maka digunakan final-activity node sebagai tanda tidak adanya aktivitas apapun/akhir suatu aktivitas.
Final-flow Node		Digunakan untuk mengakhiri aliran tertentu dalam proses.		Di samping terdapat final flow node yang menandakan akhir dari suatu opsi dalam proses. Jika tidak ada opsi lain yang diambil proses akan berhenti di sini (tidak), namun proses lain (ya) masih dapat berlanjut.
Decision Node		Digunakan untuk membuat keputusan dalam proses berdasarkan kondisi tertentu.		Pada gambar di samping, terdapat decision node yang menawarkan lebih dari satu pilihan, yaitu Pesanan ditolak atau Pesanan diterima. Ini memecah proses untuk menentukan

				kondisi mana yang sesuai dengan proses sebelumnya.
Merge Node		Menggabungkan kembali jalur keputusan yang berbeda dalam proses.		Menggabungkan kembali tindakan yang sebelumnya dipecah yaitu menutup pesanan dan mengisi pesanan.
Fork Node		Memisahkan perilaku menjadi beberapa aliran aktivitas yang berjalan secara paralel atau bersamaan.		Memisahkan tindakan menjadi beberapa bagian yaitu mengirim pesanan dan mengirim invoice yang dijalankan pada waktu yang bersamaan.
Join Node		Menyatukan kembali aliran aktivitas yang berjalan secara paralel atau bersamaan.		Menyatukan kembali tindakan mengirim pesanan dan mengirim invoice yang sebelumnya berjalan secara bersamaan.
Swimlane		Memecah diagram aktivitas menjadi bagian-bagian yang menunjukkan siapa yang bertanggung		Dosen merupakan penanggung jawab pertama pada proses pengajaran soal seperti membuat

		<p>jawab atas setiap langkah dalam proses.</p>		<p>soal, menilai jawaban, dsb. Sedangkan mahasiswa merupakan sebaliknya (seperti gambar di samping).</p>
--	--	--	--	--

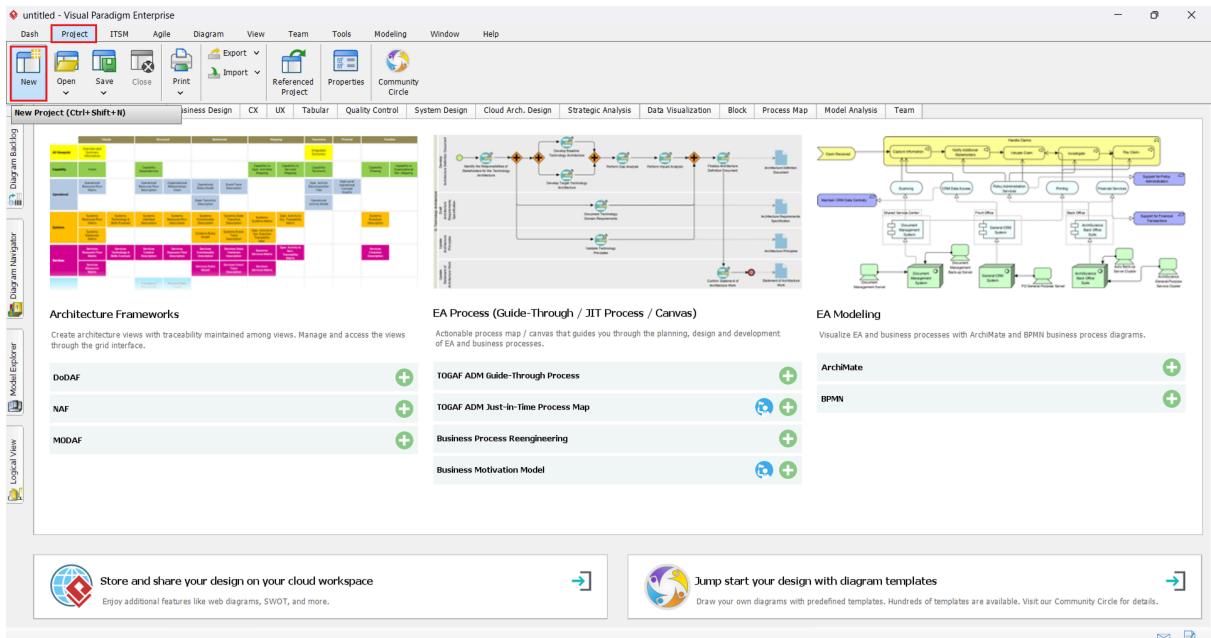
## LANGKAH-LANGKAH PRAKTIKUM

### A. Cara Mendokumentasikan Functional Requirement

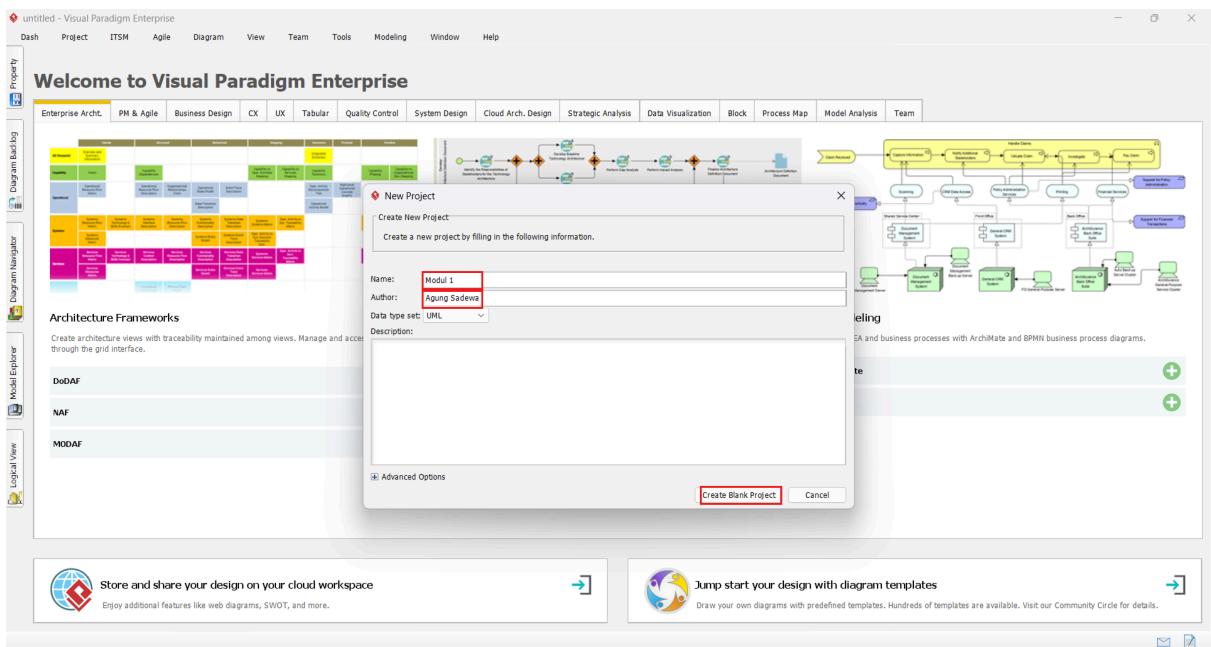
No	Aktor	Deskripsi Aktor	<i>Functional Requirements</i>
1.	[Nama Aktor] <b>Contoh: Pelanggan</b>	Deskripsi mengenai fungsi atau peranan aktor terhadap aplikasi. <b>Contoh: Aktor sebagai pengguna aplikasi yang melakukan pemesanan makanan</b>	1. [Nama Aktor] dapat melakukan registrasi
			2. [Nama Aktor] dapat melakukan login
			3. [Nama Aktor] dapat memilih menu
			4. ...
2.	...	...	1. ...

### B. Cara Membuat Use Case Diagram

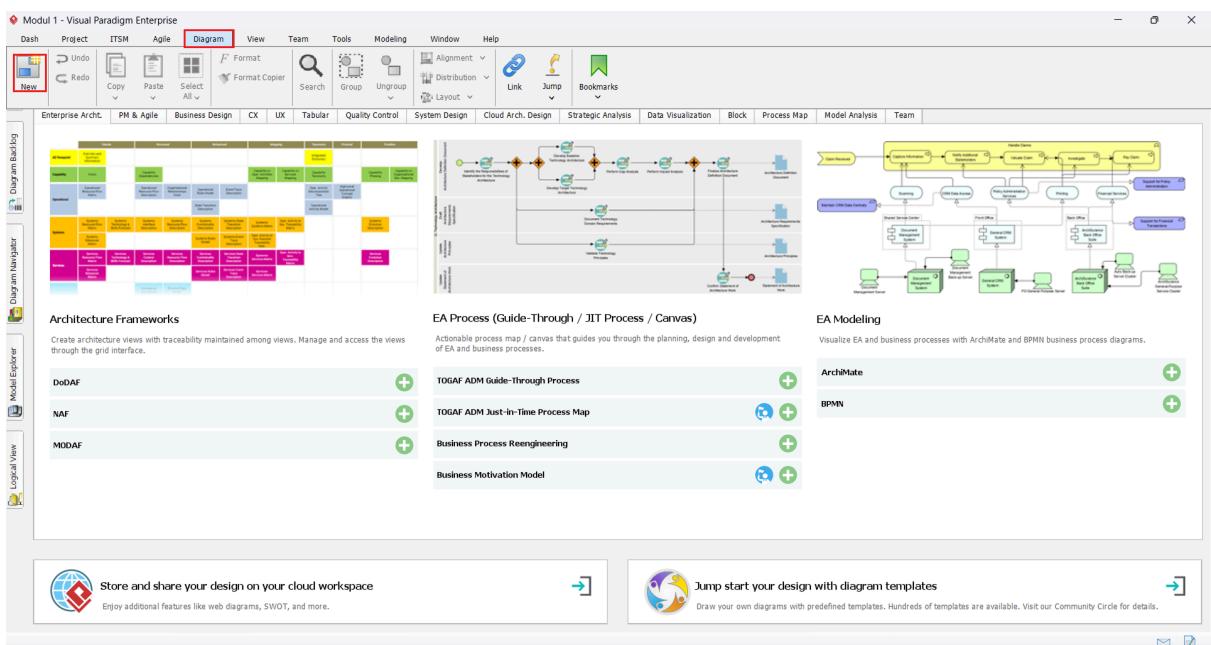
- Pergi ke tab project dibagian atas kemudian klik new.



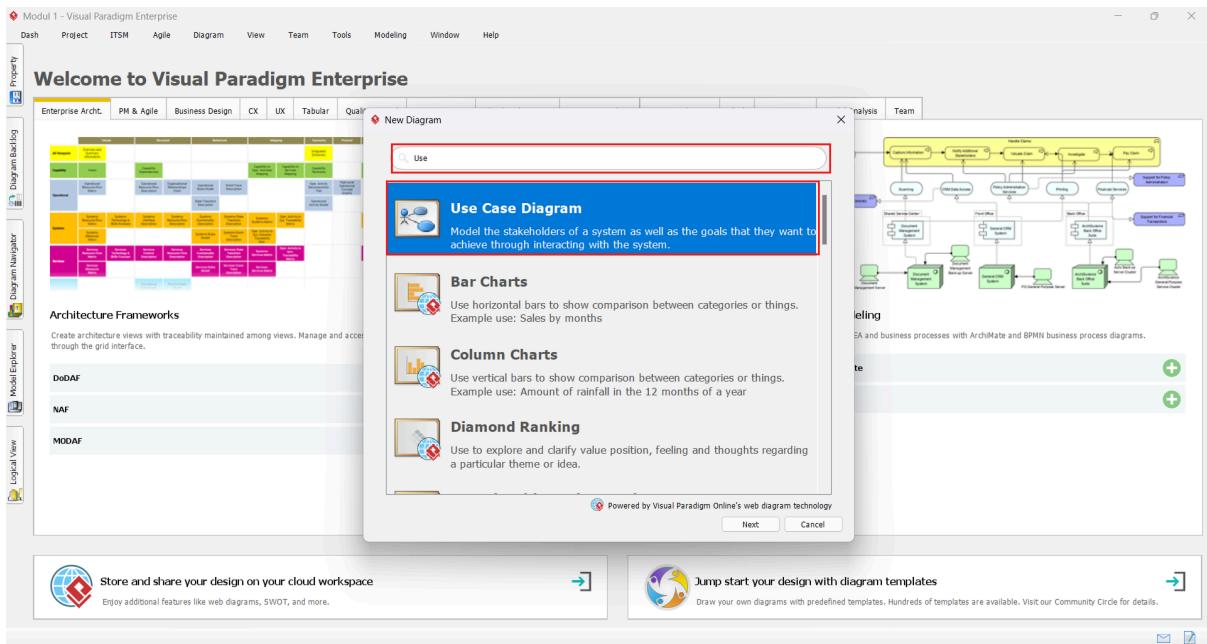
2. Masukkan nama project beserta modul yang dikerjakan, serta cantumkan nama pembuat project pada bagian author.



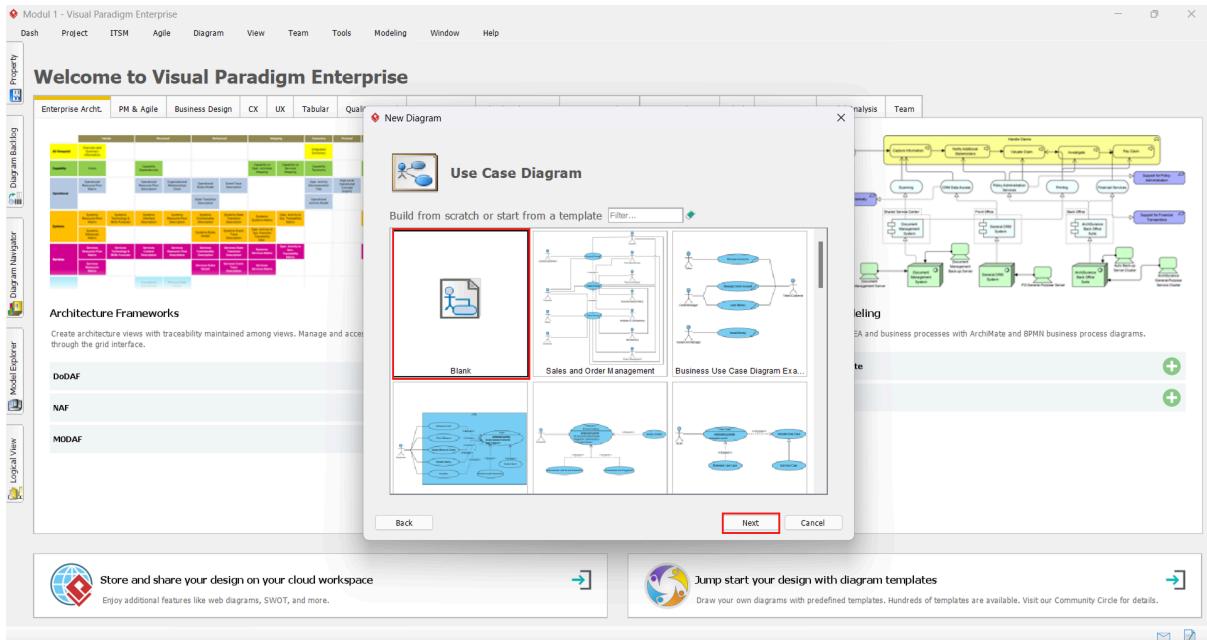
3. Setelah masuk ke project yang dibuat pilih tab diagram, kemudian klik new.



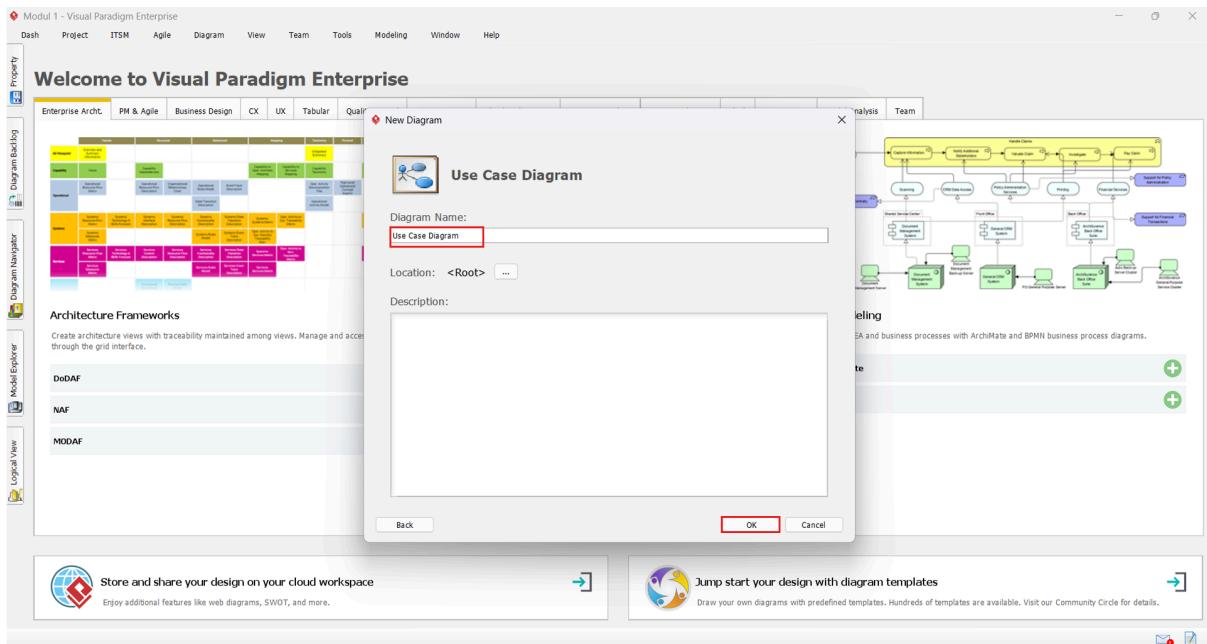
4. Cari dengan kata kunci “Use Case Diagram” lalu klik diagram yang ingin dibuat.



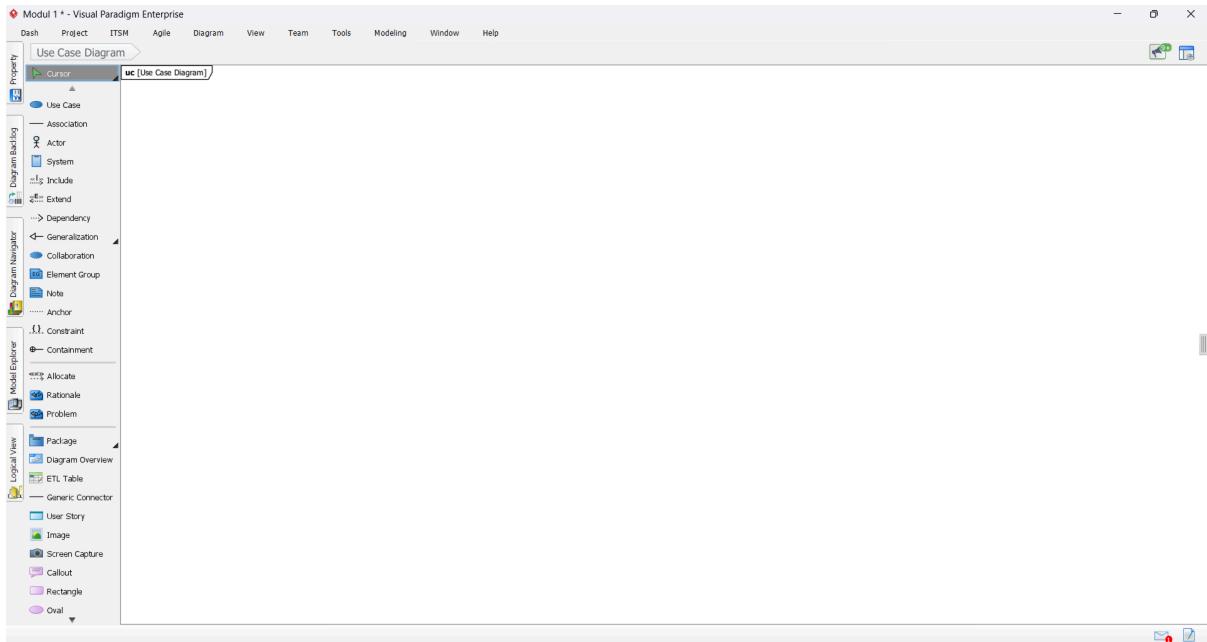
5. Pilih template “Blank” lalu klik next.



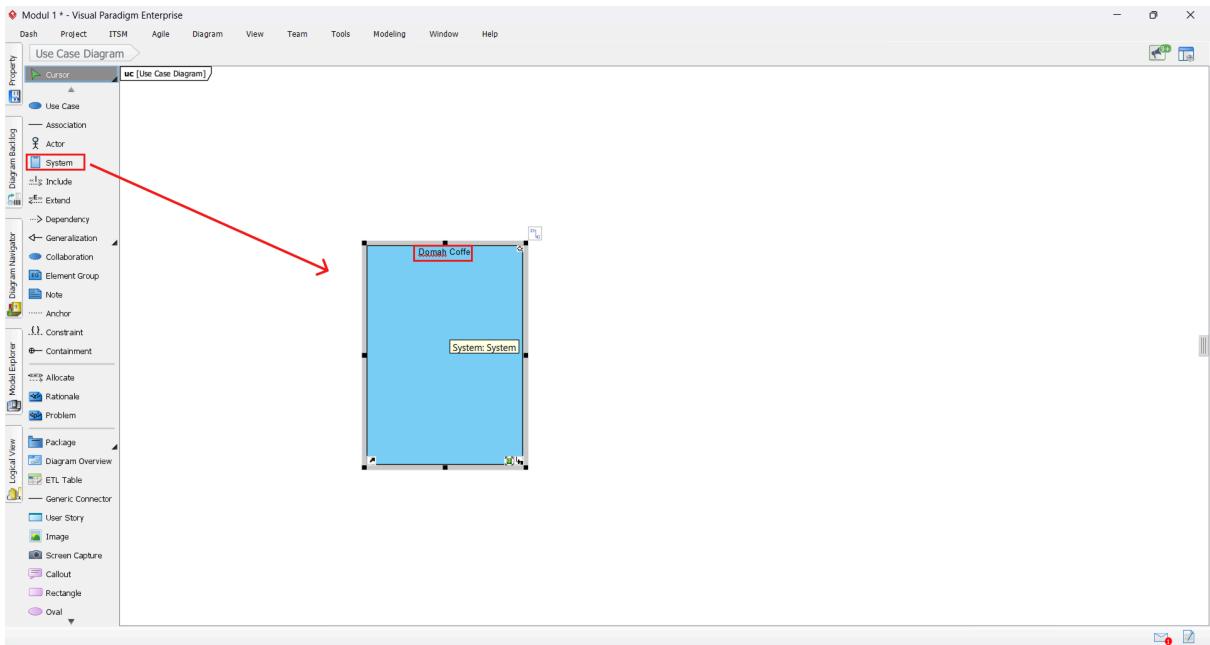
6. Selanjutnya, masukkan nama untuk diagram yang baru dibuat. Setelah itu, klik OK.



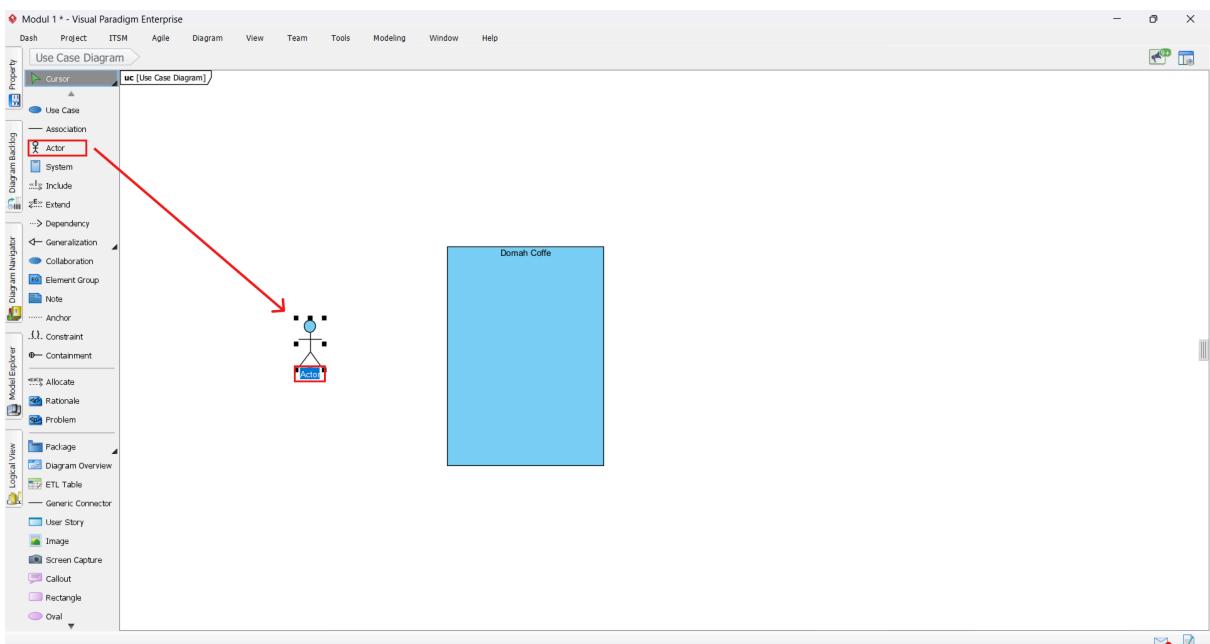
7. Tampilan ketika diagram sudah berhasil terbuat.



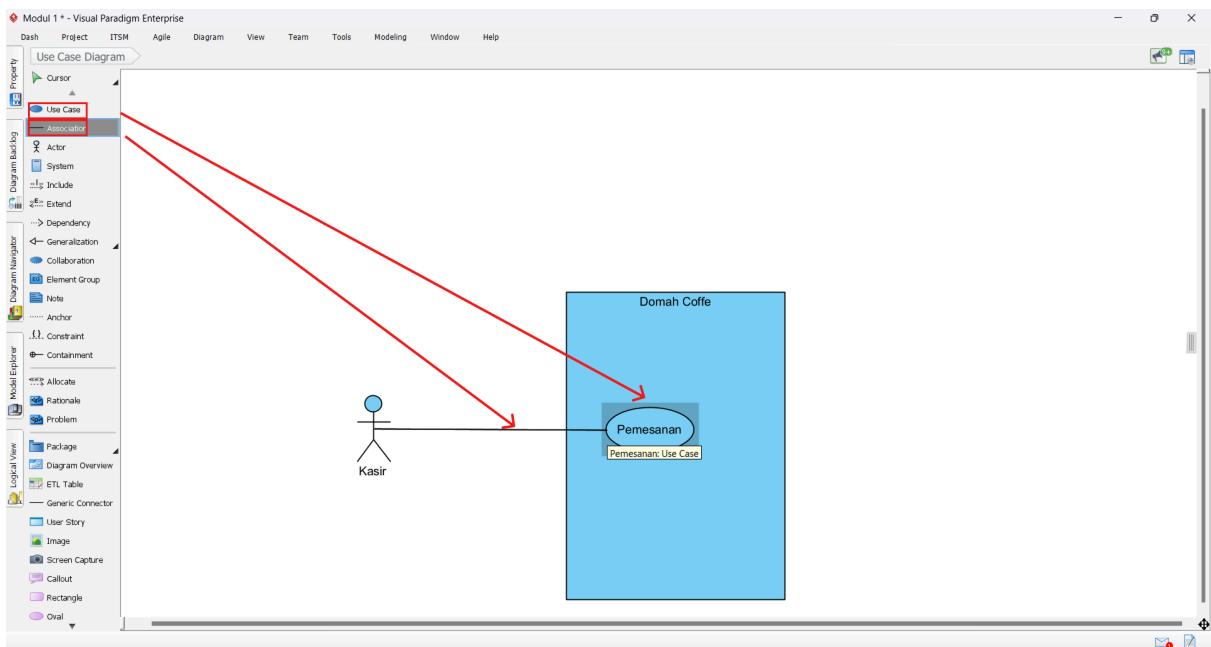
8. Klik ikon "system" kemudian di drag untuk mulai membuat *use case diagram*, lalu double-click pada judul system di atas untuk mengganti namanya.



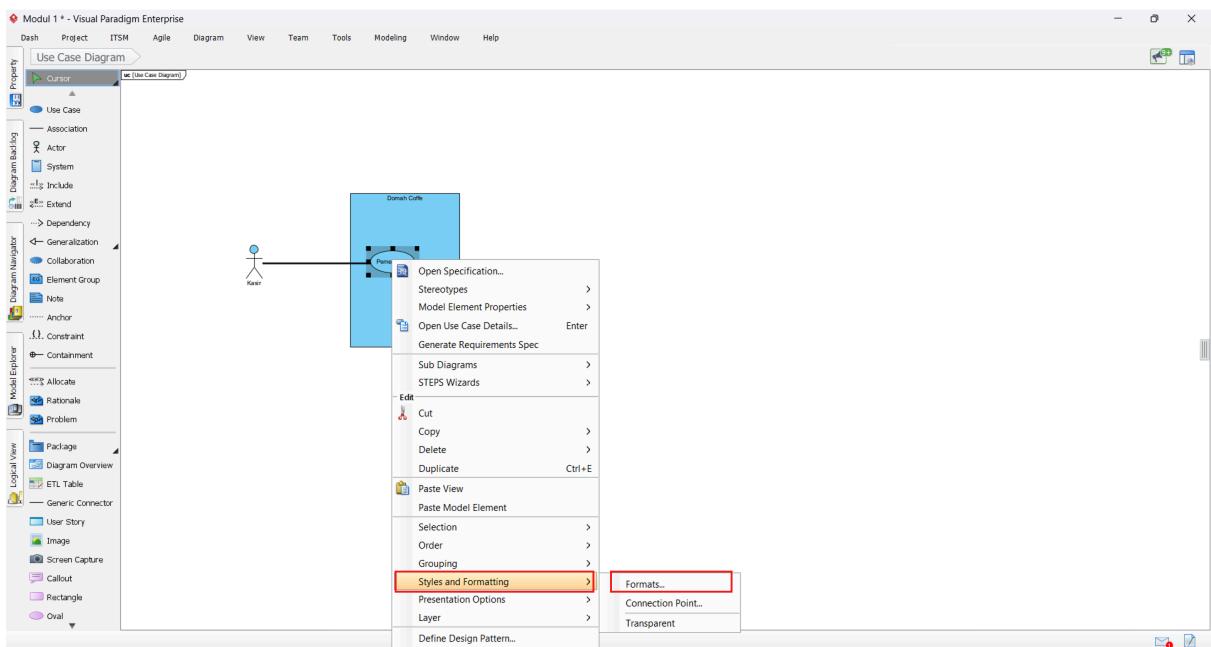
9. Selanjutnya, buat actor dengan mengklik ikon aktor pada bagian notasi, lalu drag dan *double-click* untuk mengganti namanya.



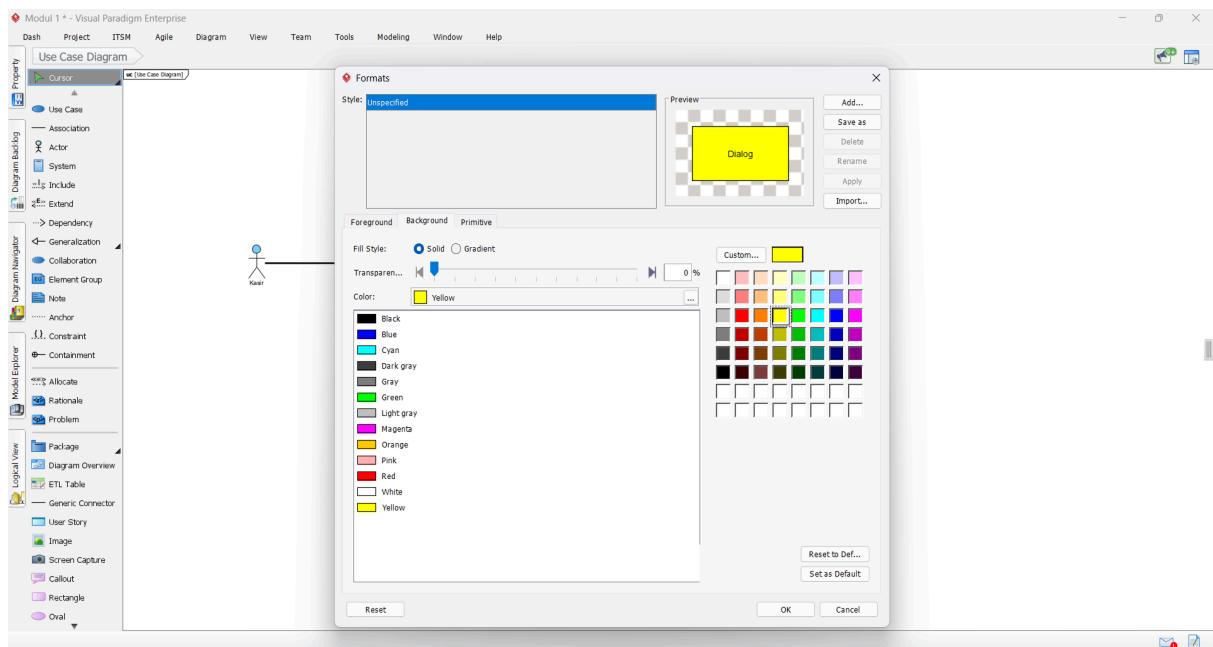
10. Selanjutnya, untuk membuat *use case*, klik "Use Case" pada bagian notasi, lalu drag ke dalam sistem dan isi nama *use case* sesuai dengan studi kasus. Kemudian, pilih actor, lalu drag ke *use case* yang ingin diasosiasikan atau dihubungkan dengan actor.



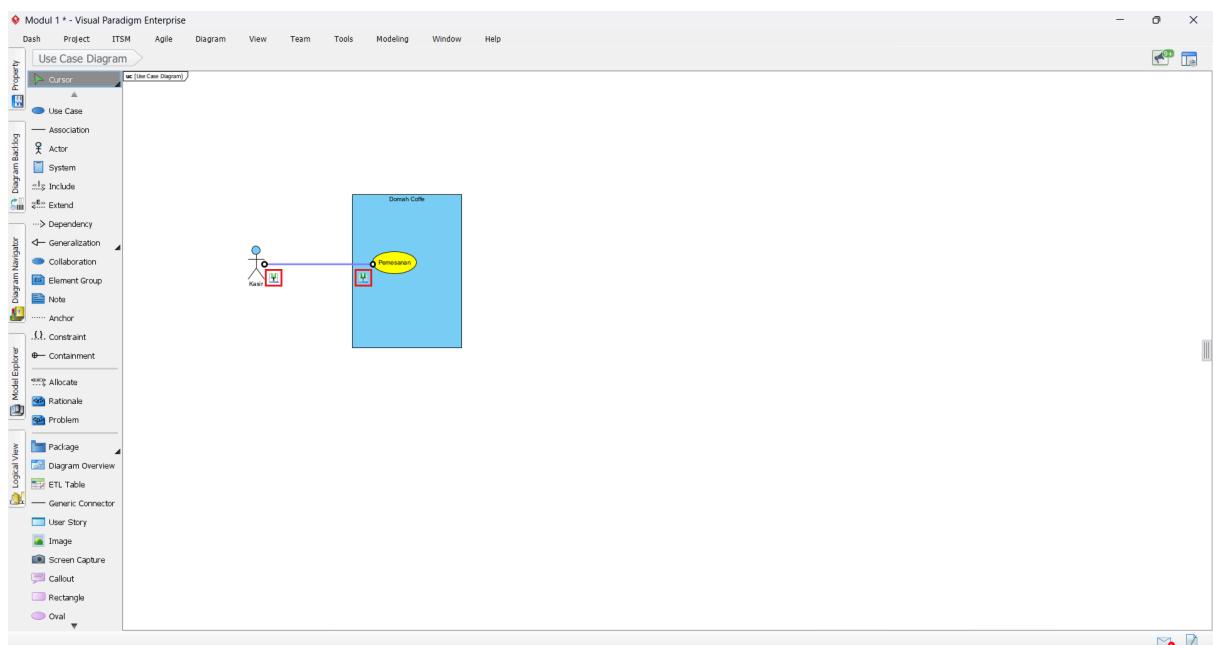
11. Untuk mengganti warna pada *use case*, klik *use case* yang ingin diubah warnanya, lalu klik kanan, pilih Styles and Formatting, kemudian klik Formats.



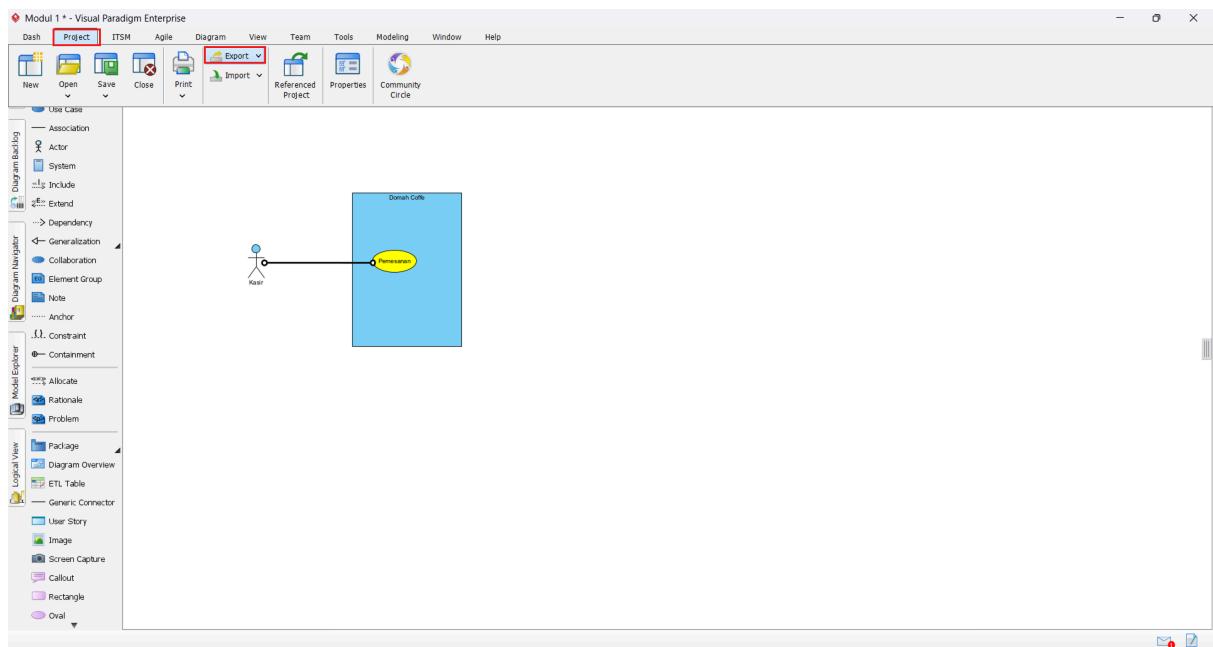
12. Setelah itu, pilih warna sesuai keinginan, lalu klik OK.



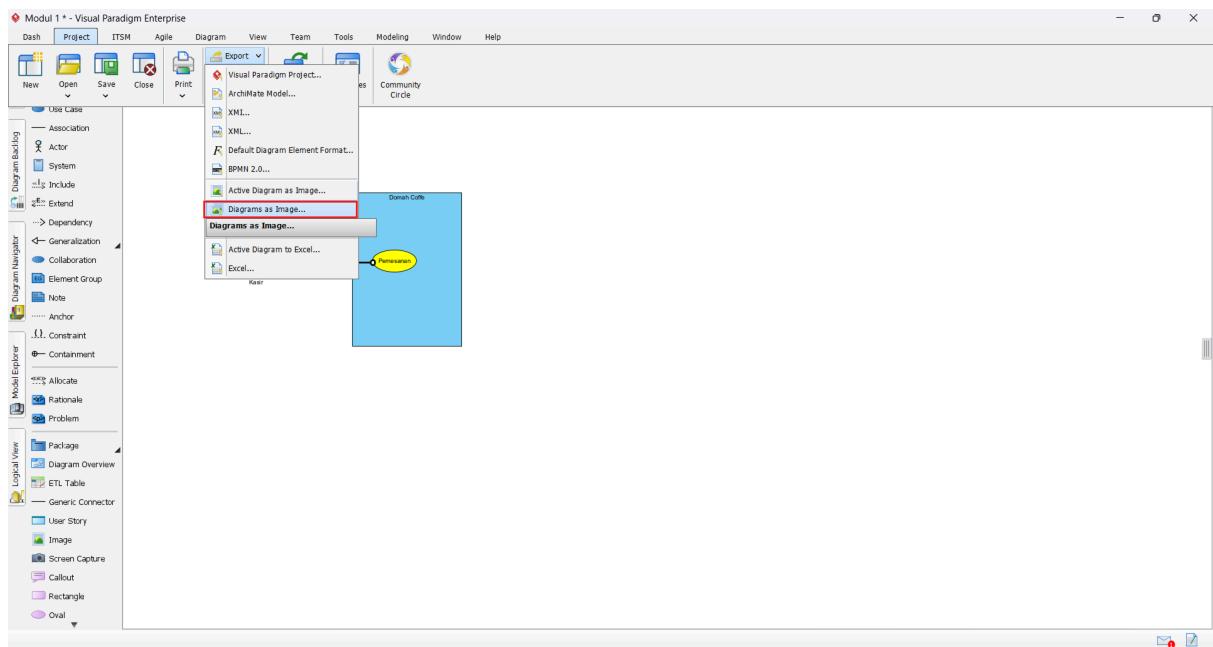
13. Untuk memudahkan dalam menggeser aktor dan Use Case, klik bagian Association, lalu tekan Pin pada elemen yang ingin dikunci.



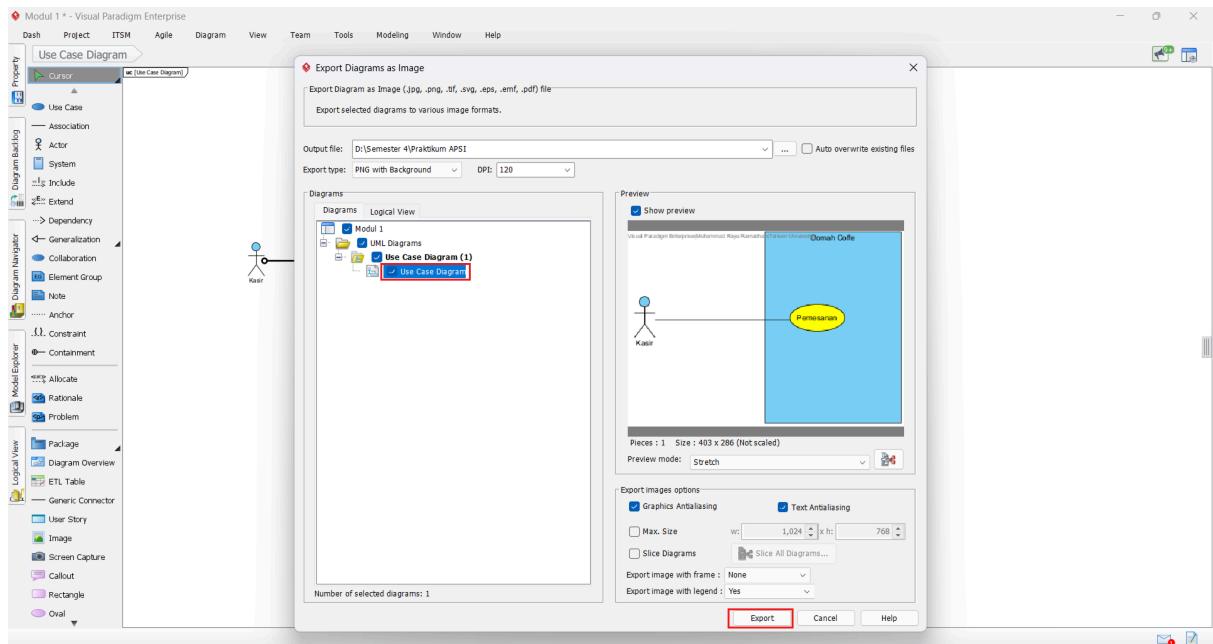
14. Untuk mengekspor proyek Use Case Diagram, buka tab Project, lalu klik Export.



15. Setelah itu, pilih diagram as image.

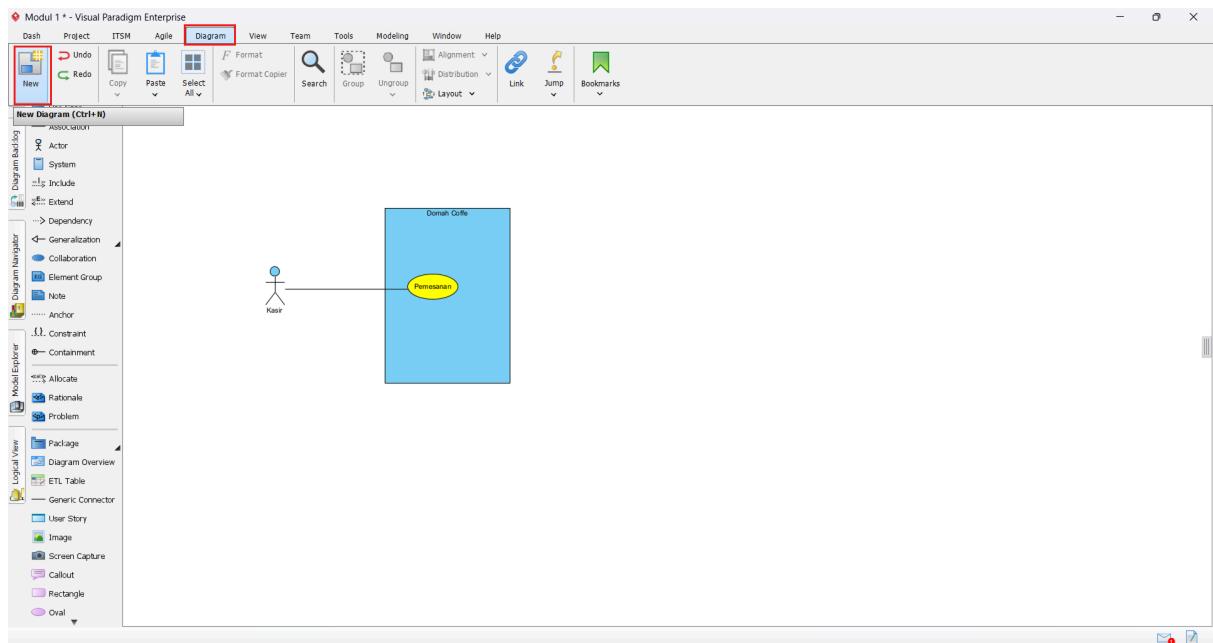


16. Lalu, centang diagram yang ingin diekspor, tentukan lokasi penyimpanan file, dan klik tombol Export.

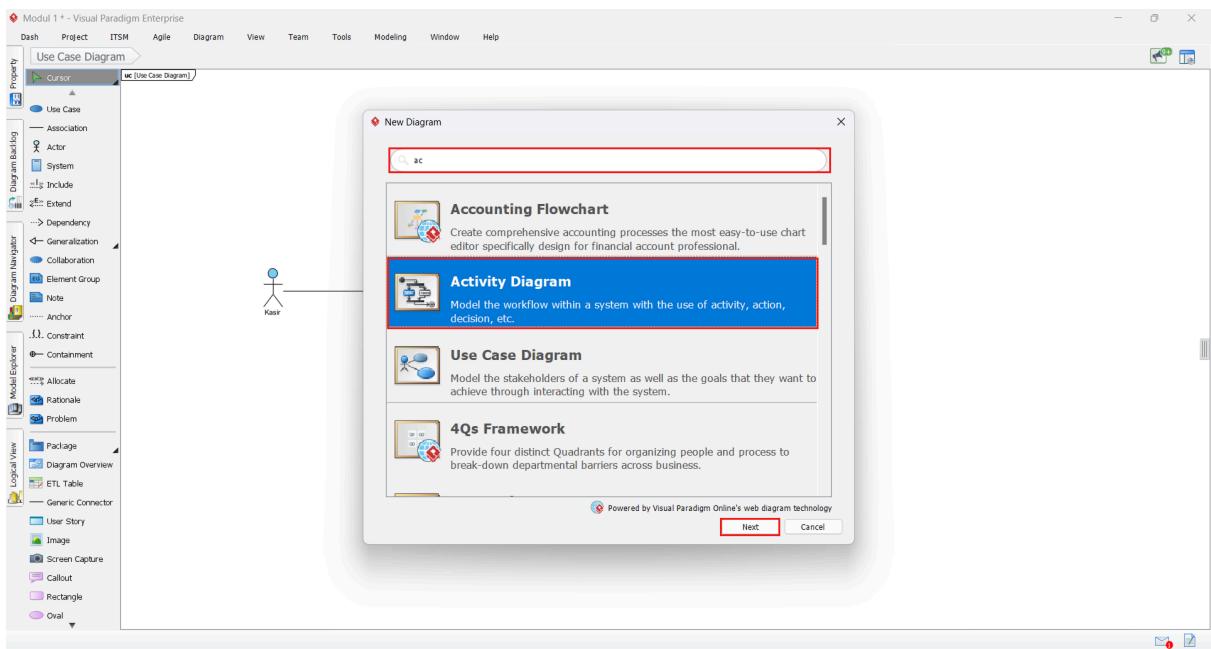


### C. Cara Membuat Activity Diagram

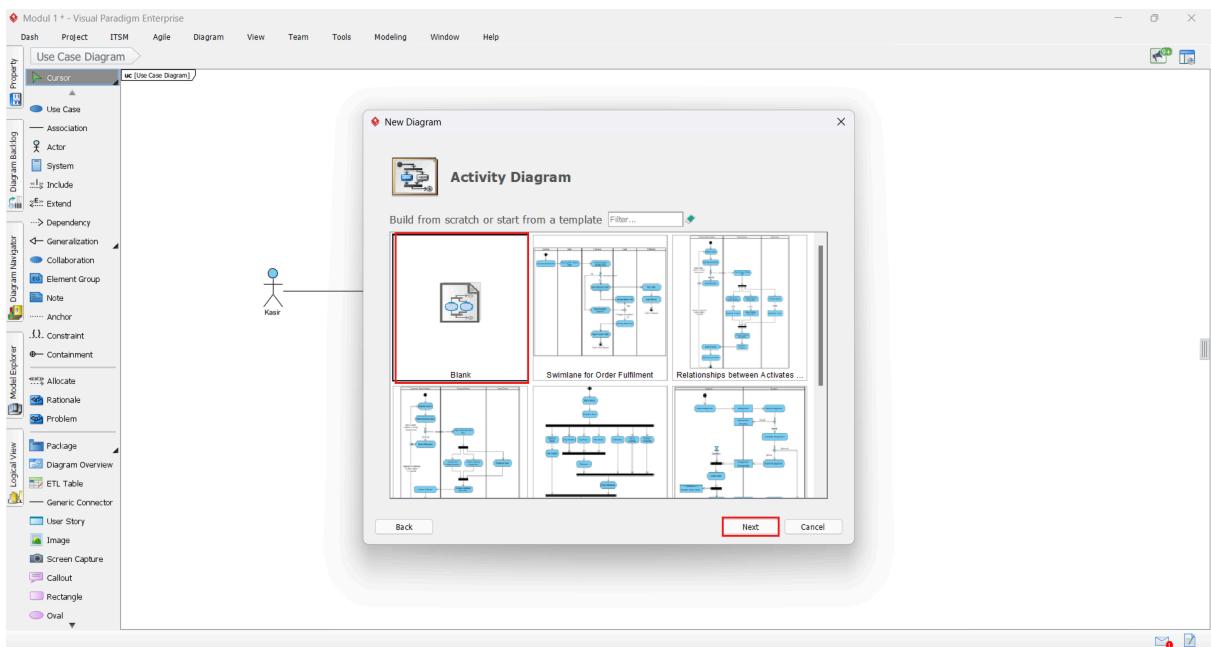
1. Masuk ke tab diagram kemudian klik new



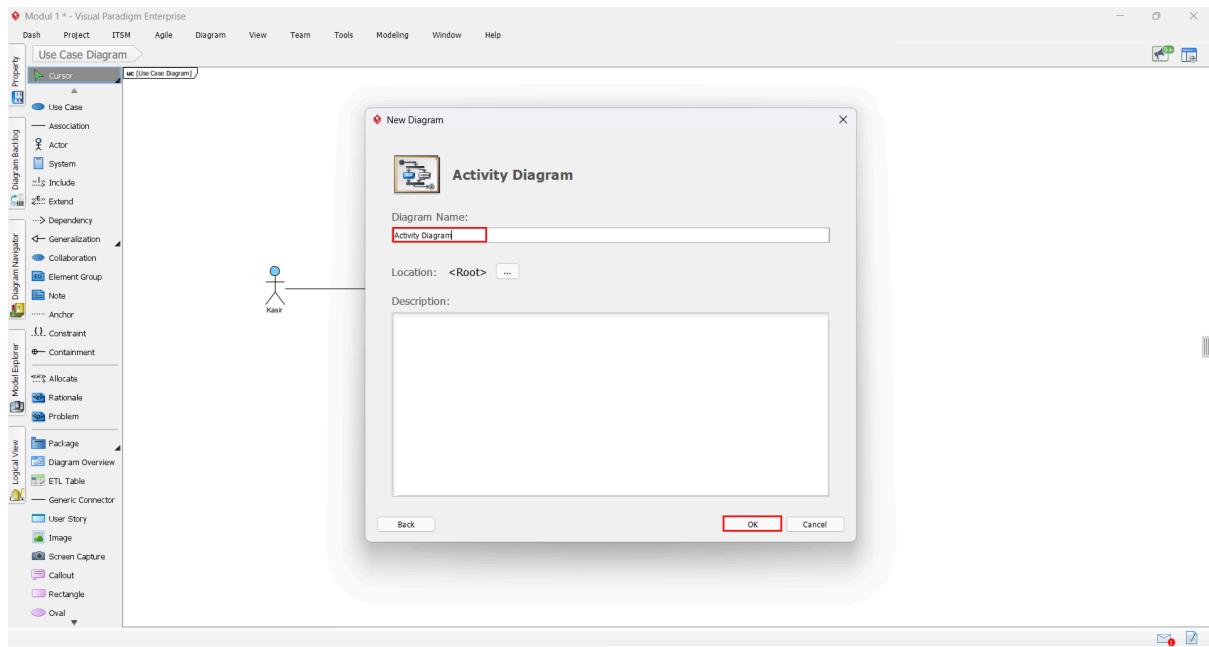
2. Cari dengan kata kunci “Activity Diagram” lalu klik next.



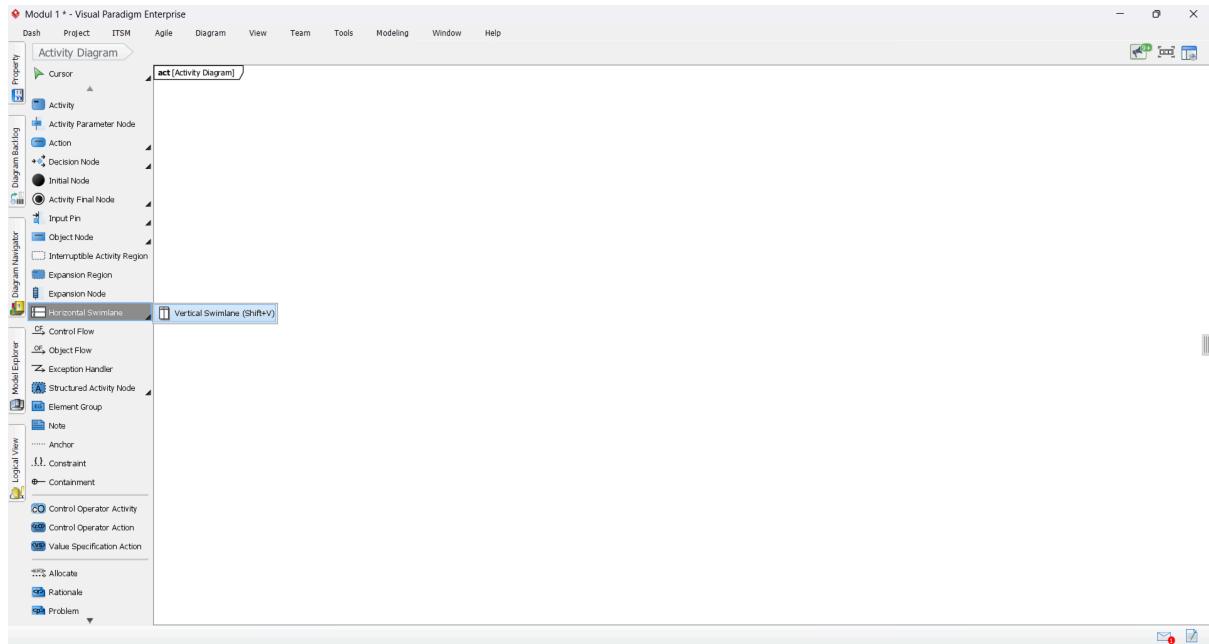
3. Pilih template “Blank” lalu klik next.



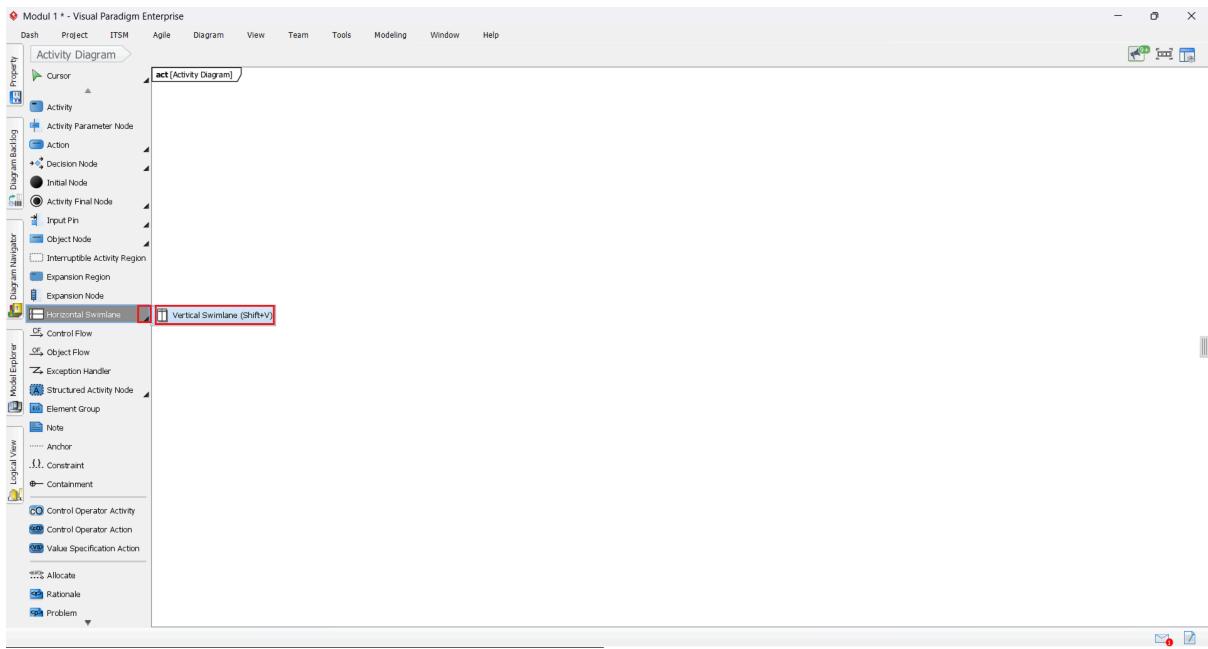
4. Selanjutnya, masukkan nama untuk diagram yang baru dibuat. Setelah itu, klik OK.



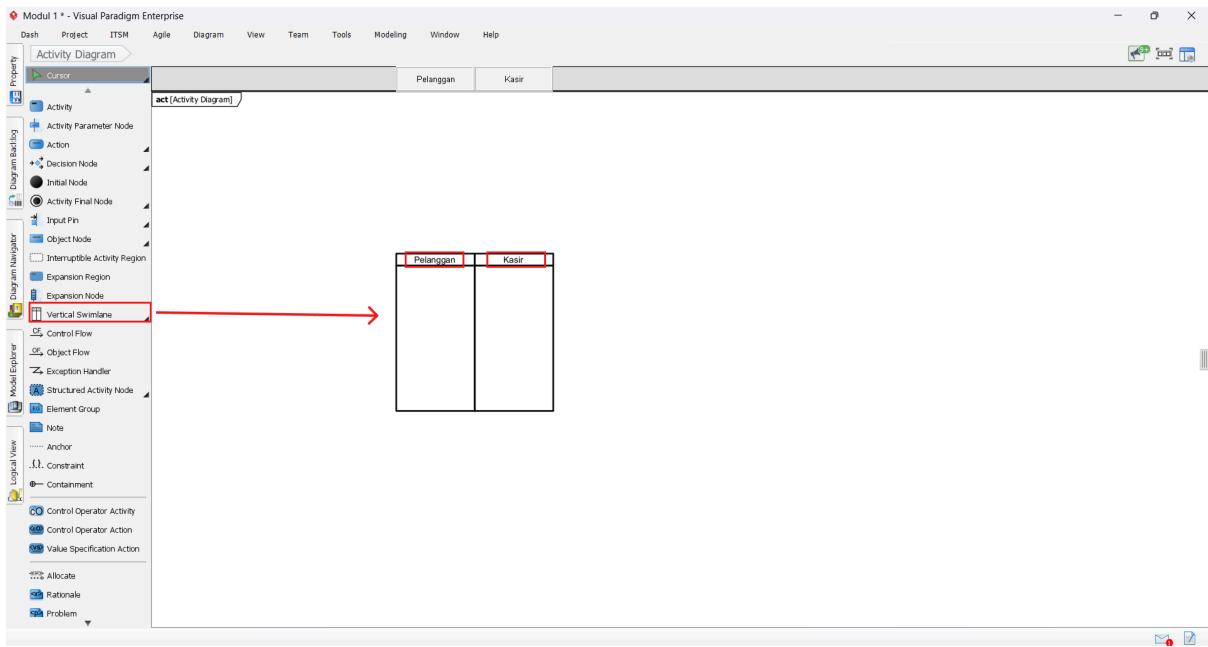
5. Tampilan ketika diagram sudah berhasil terbuat.



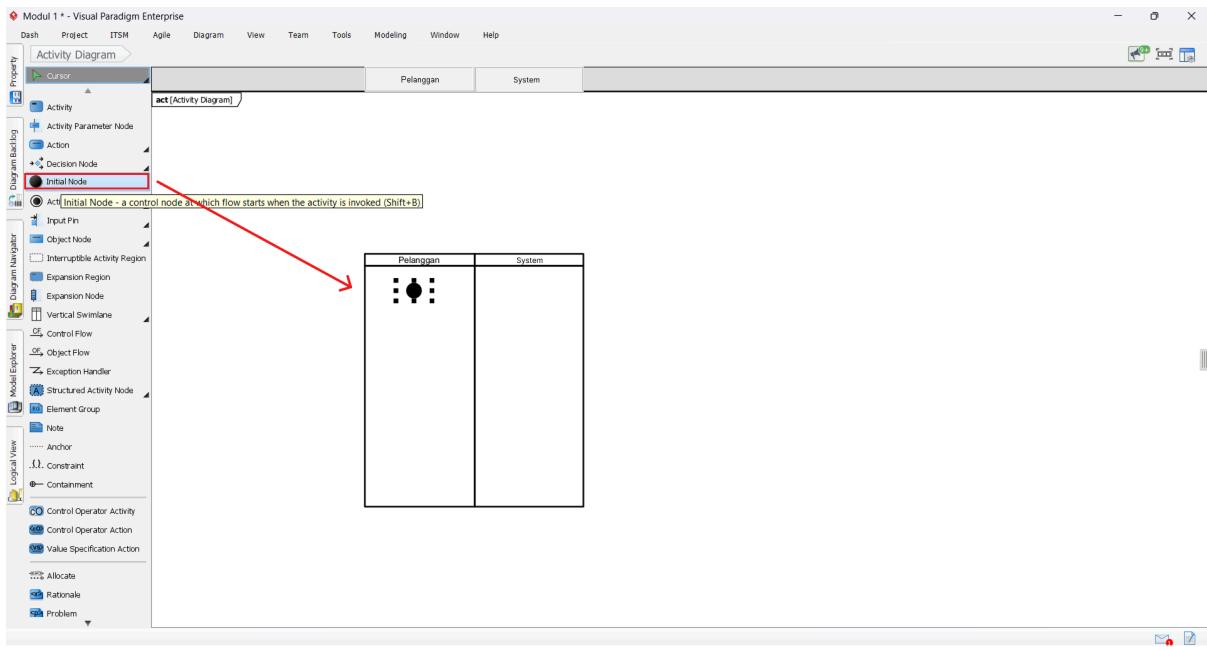
6. Pilih Icon Swimlane, kemudian klik tombol hitam di pojok kanan lalu ubah ke vertical swimlane



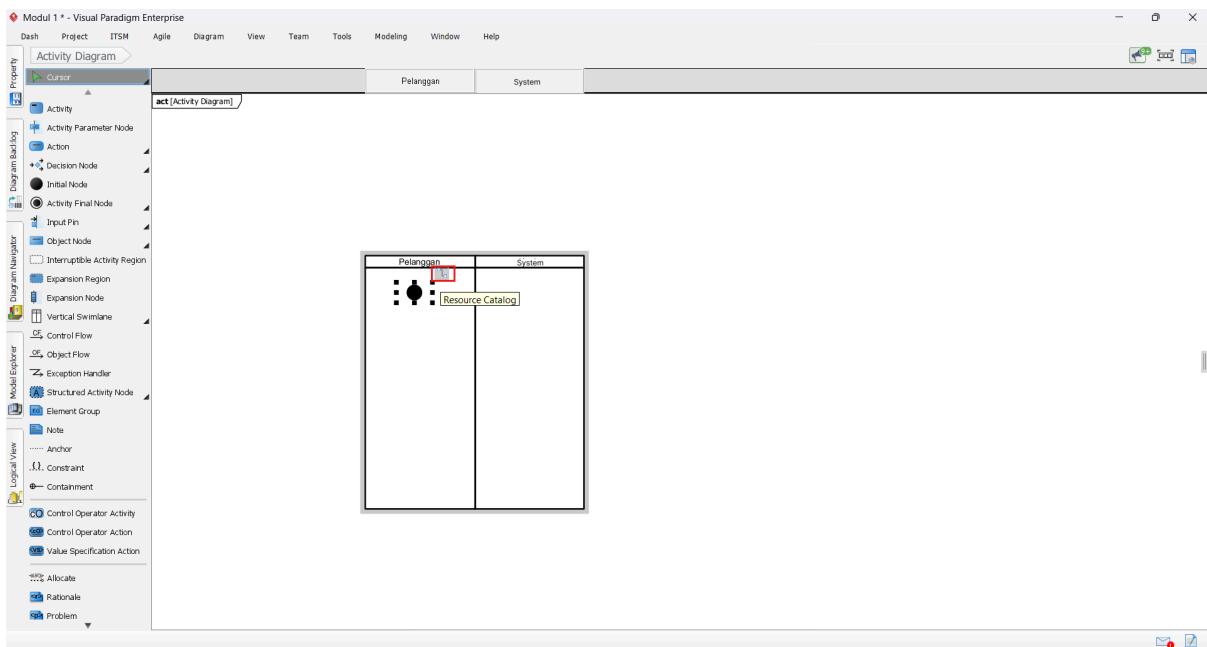
7. Klik ikon Swimlane, lalu drag ke area kerja. Setelah itu, double-click pada bagian atas Swimlane dan ubah namanya sesuai kebutuhan atau studi kasus.



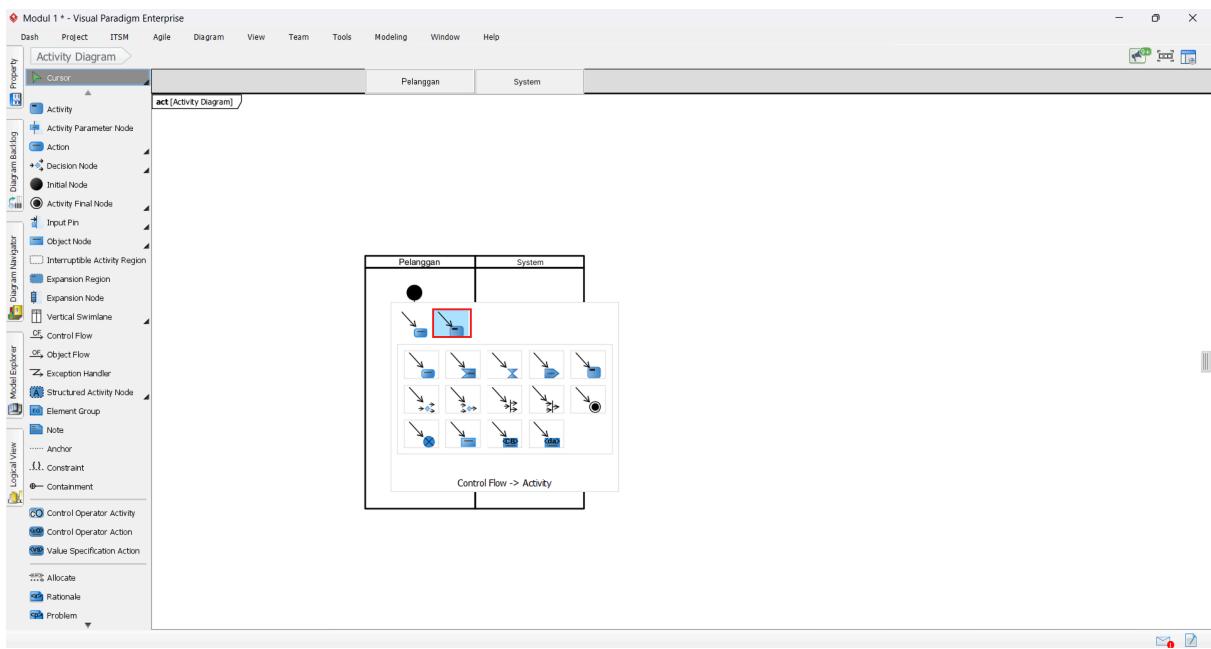
8. Klik ikon “Initial Node” pada bagian notasi, lalu tarik dan letakkan (drag and drop) ke dalam Swimlane.



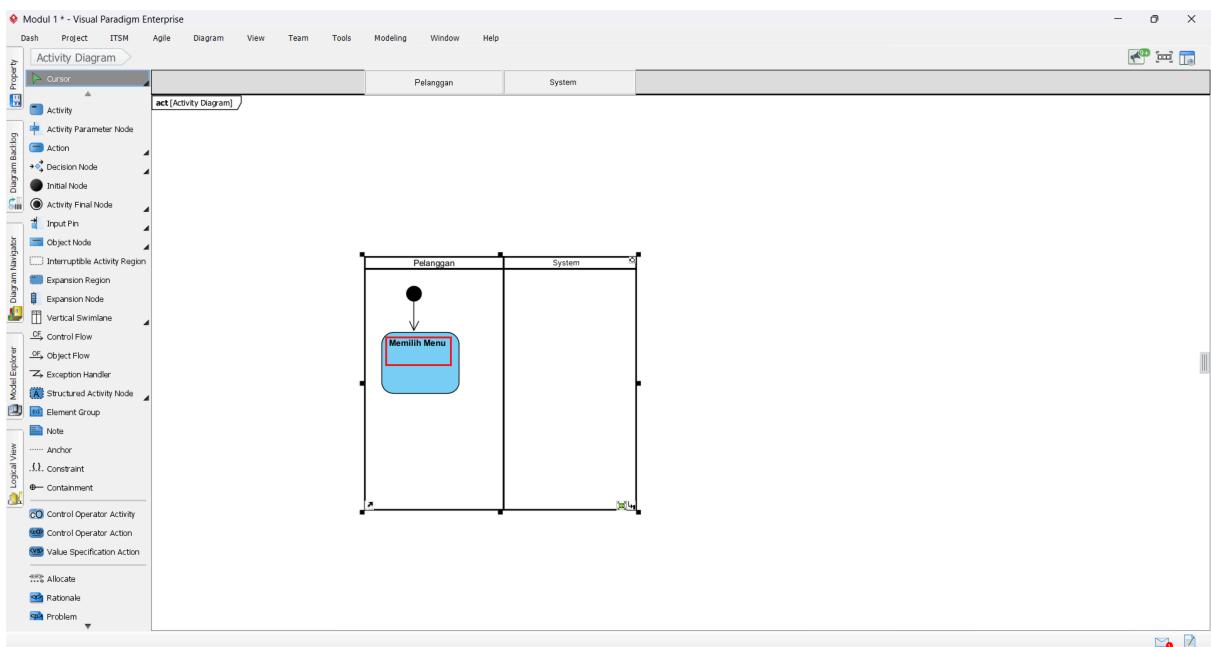
9. Kemudian, klik Initial Node yang berada di dalam Swimlane, lalu klik Resource Catalog yang muncul di kanan atas notasi tersebut.



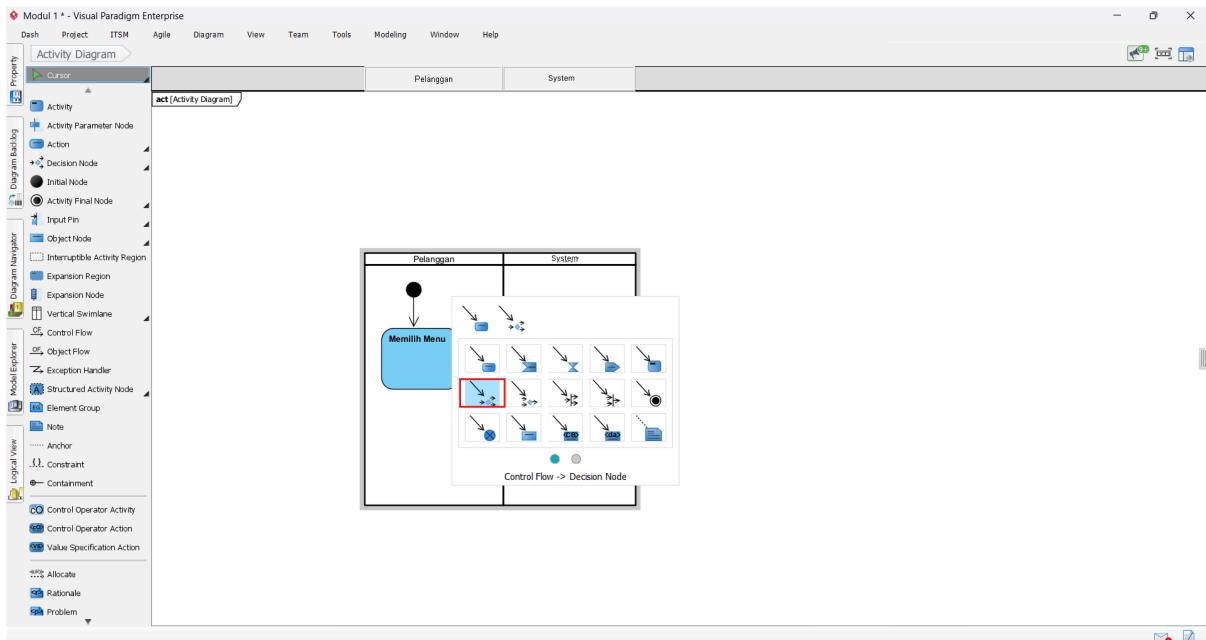
10. Pilih notasi Activity, lalu klik dan letakkan di kolom Swimlane sesuai dengan urutan alur yang diinginkan.



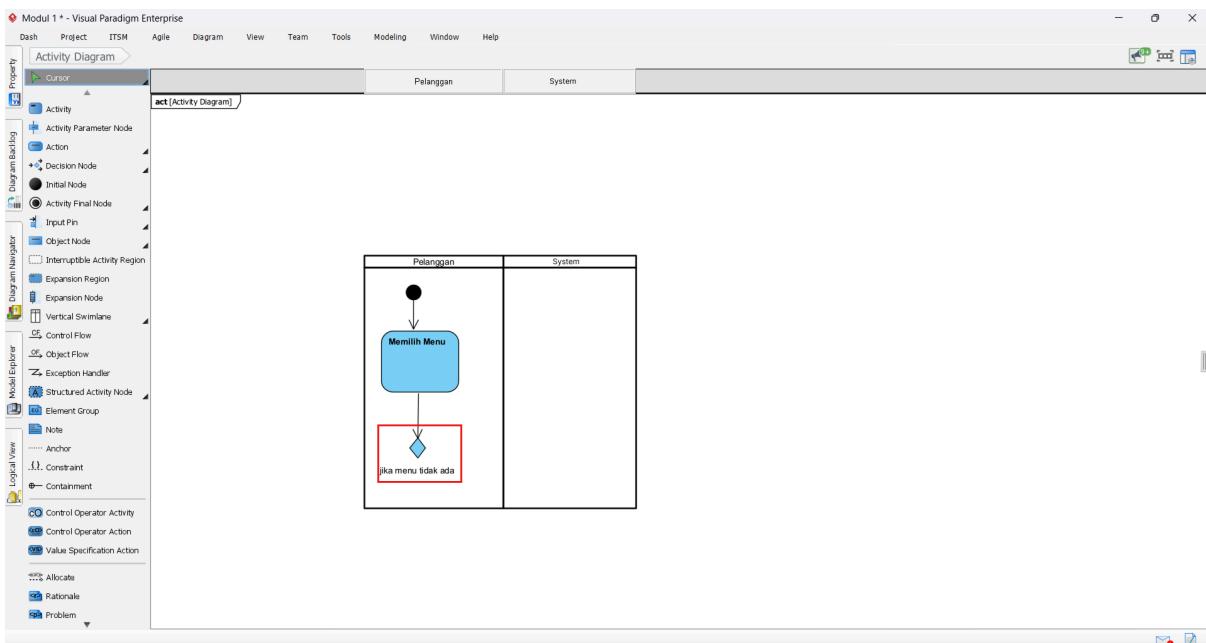
11. Lakukan *double-click* pada Activity untuk mengganti namanya sesuai kebutuhan.



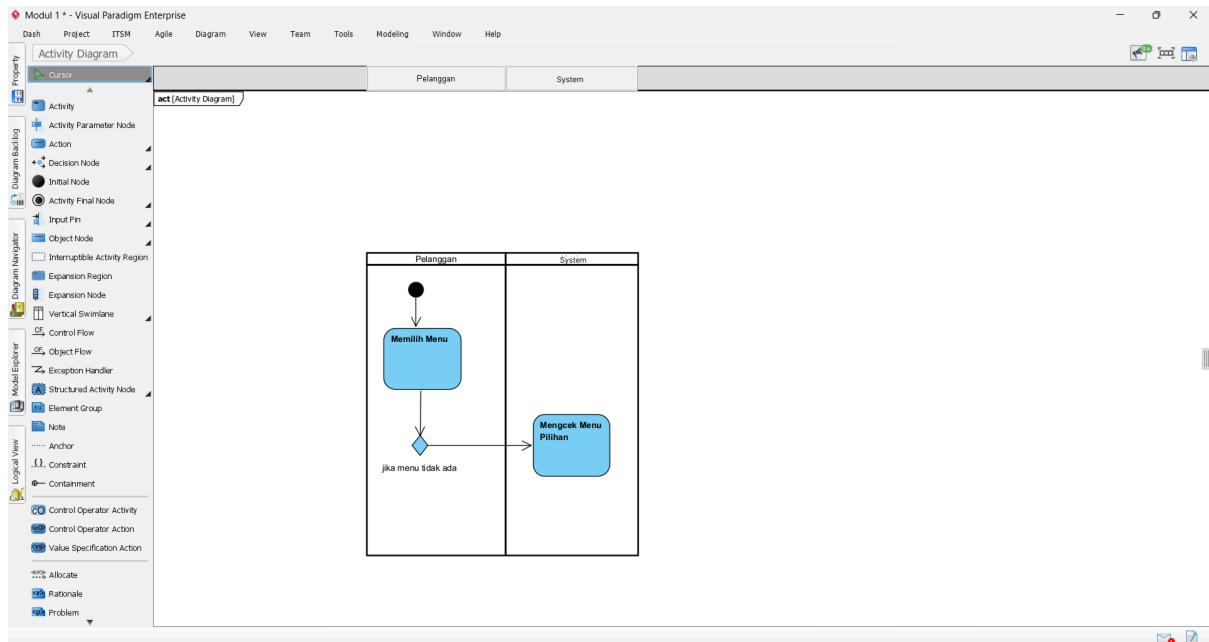
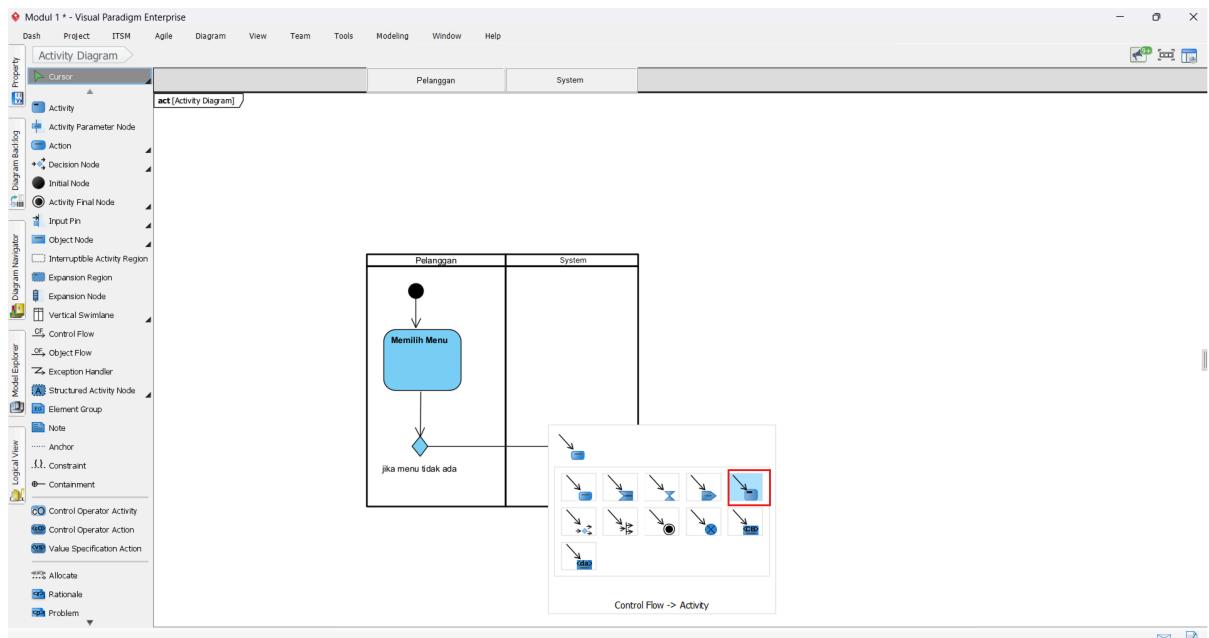
12. Jika ingin menggunakan Decision Node, klik Activity, lalu klik Resource Catalog di sebelah kanan atas notasi. Setelah itu, pilih notasi Decision Node, kemudian klik dan sesuaikan posisinya dengan alur yang telah dibuat.



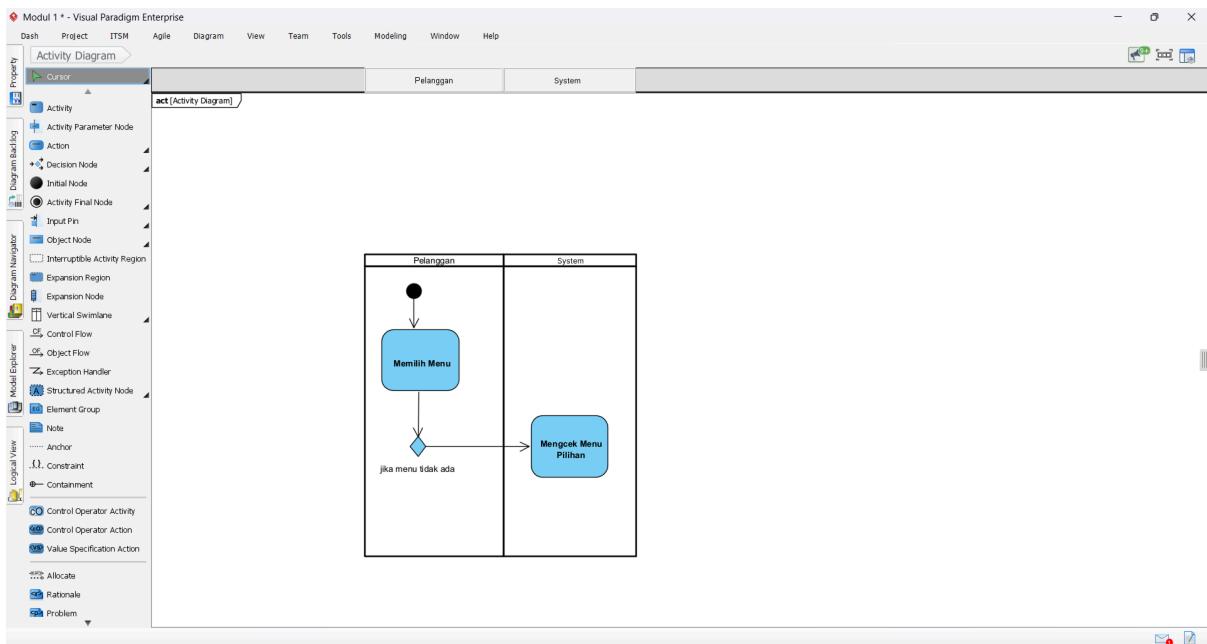
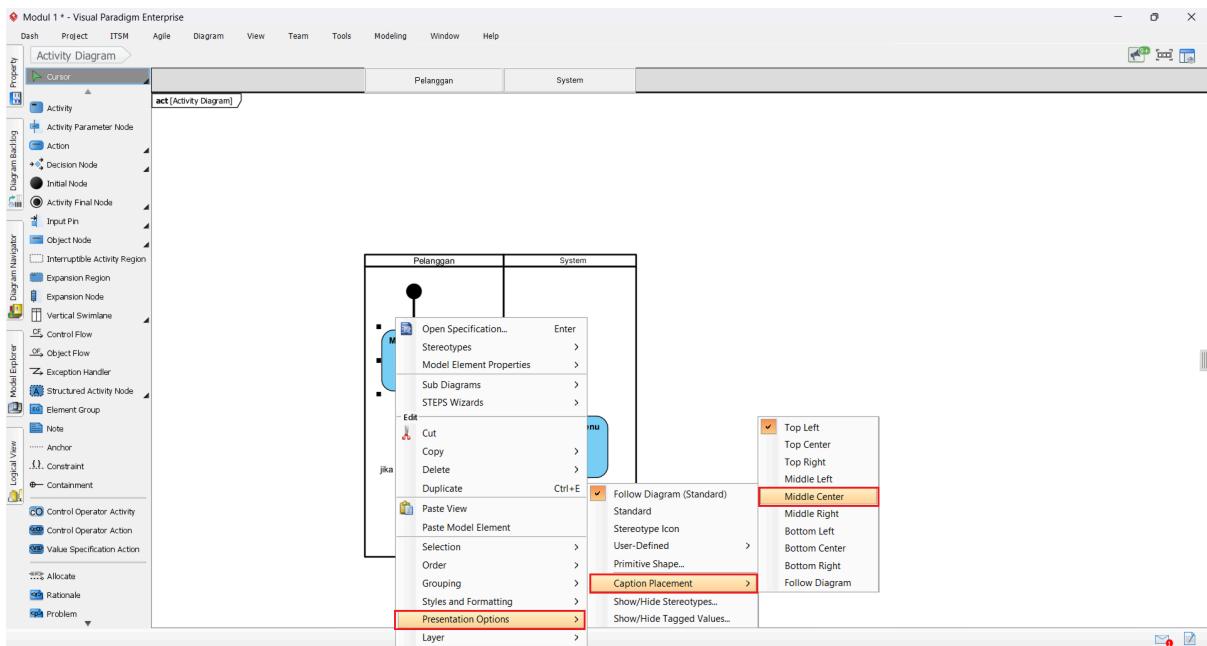
13. Untuk mengisi Decision sesuai dengan studi kasus, lakukan double-click pada notasi Decision yang berada di dalam Swimlane.



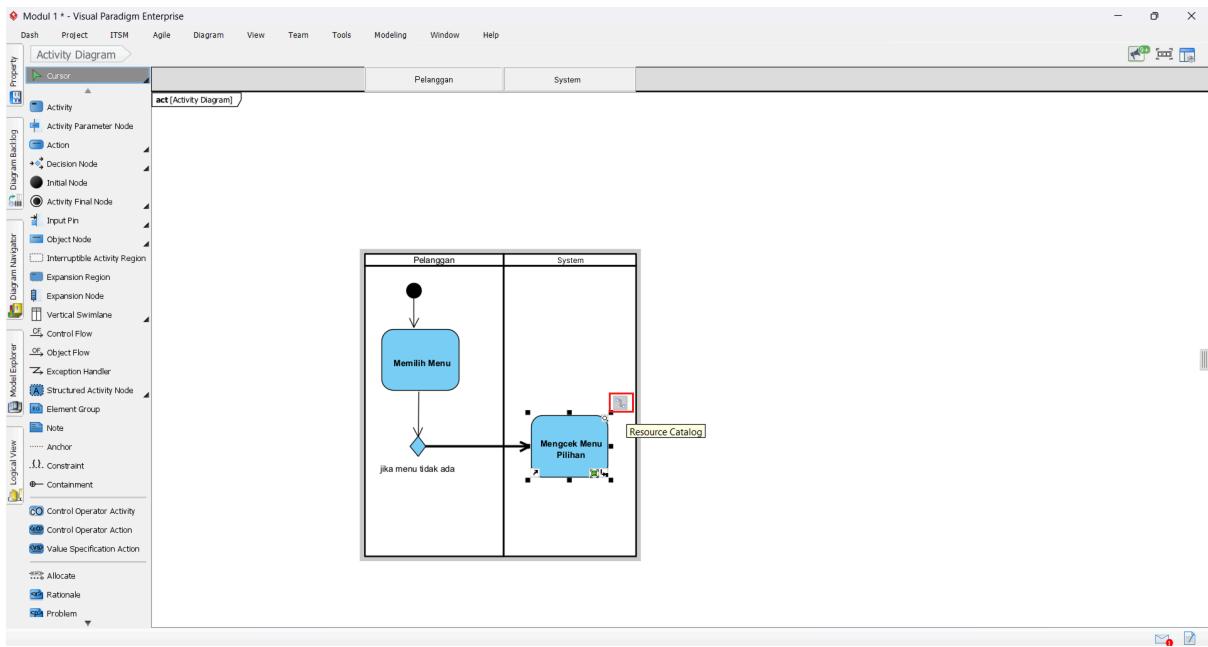
14. Klik *Decision Node*, lalu klik *Resource Catalog* di sebelah kanan atas notasi. Setelah itu, pilih notasi *Activity*, lalu klik dan tempatkan sesuai dengan alur yang telah dibuat.



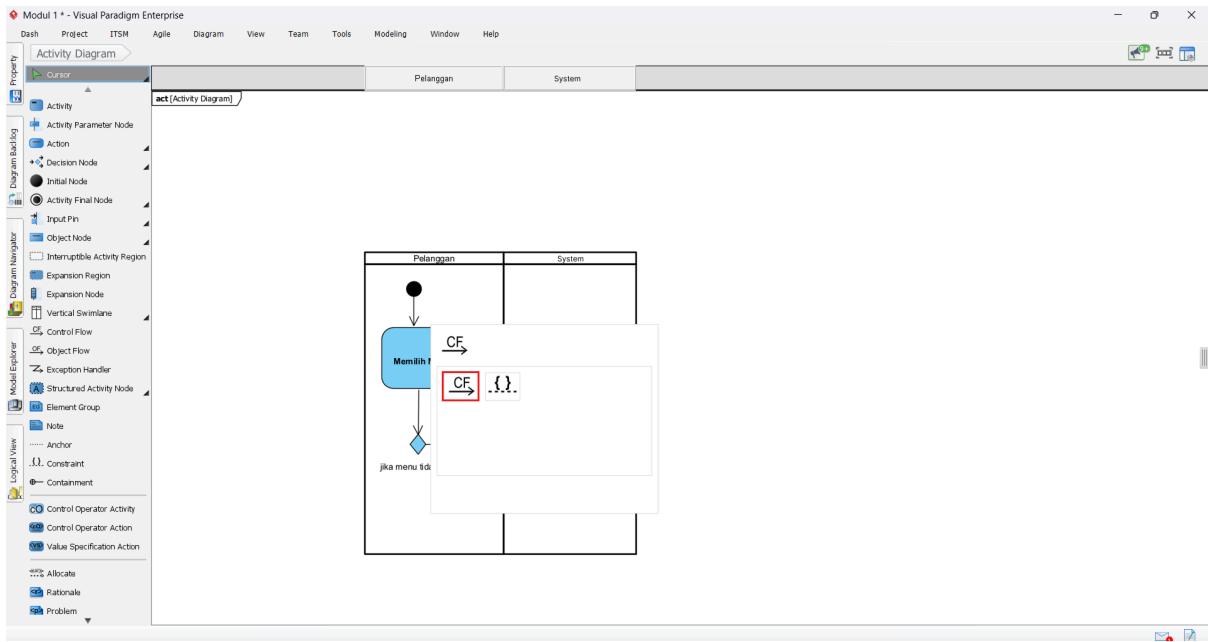
15. Jika ingin mengubah posisi teks di dalam Activity agar berada di tengah, klik notasi Activity, lalu klik kanan dan pilih Presentation Options. Setelah itu, klik bagian Caption Placement, kemudian pilih Middle Center.

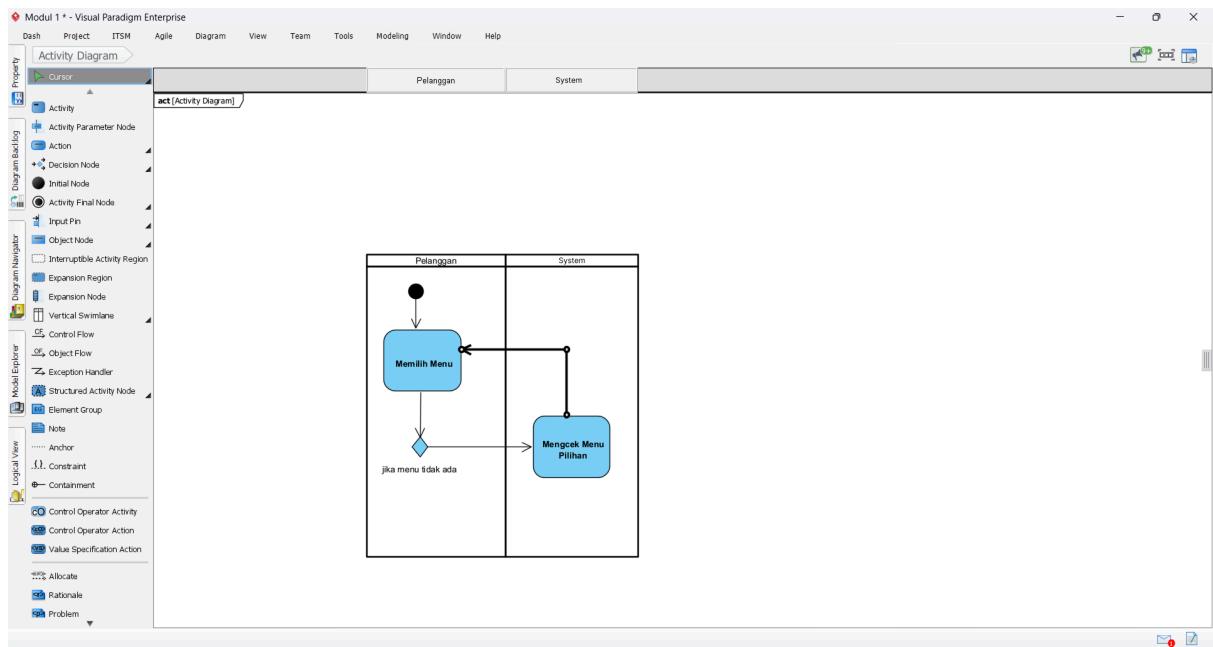


16. Selanjutnya klik aktiviy yang berada di dalam Swimlane, lalu klik Resource Catalog yang muncul di kanan atas notasi tersebut.

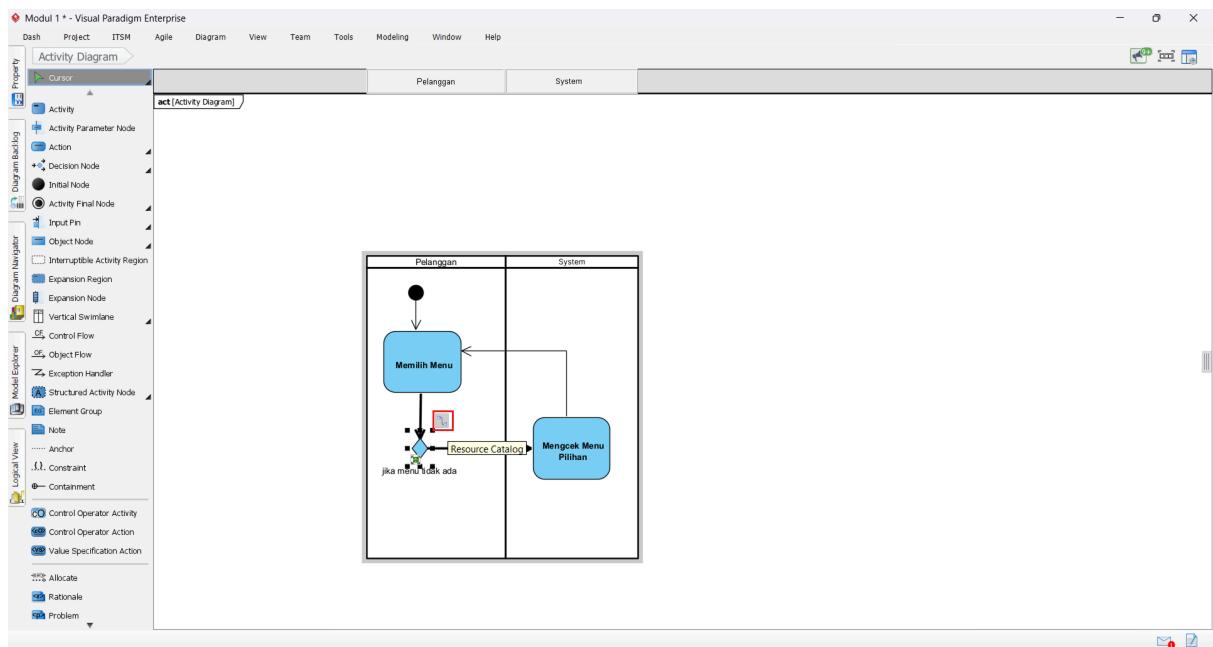


17. Pilih notasi Control Flow, lalu klik dan arahkan garis sesuai dengan alur yang telah dibuat untuk menghubungkan elemen-elemen diagram.

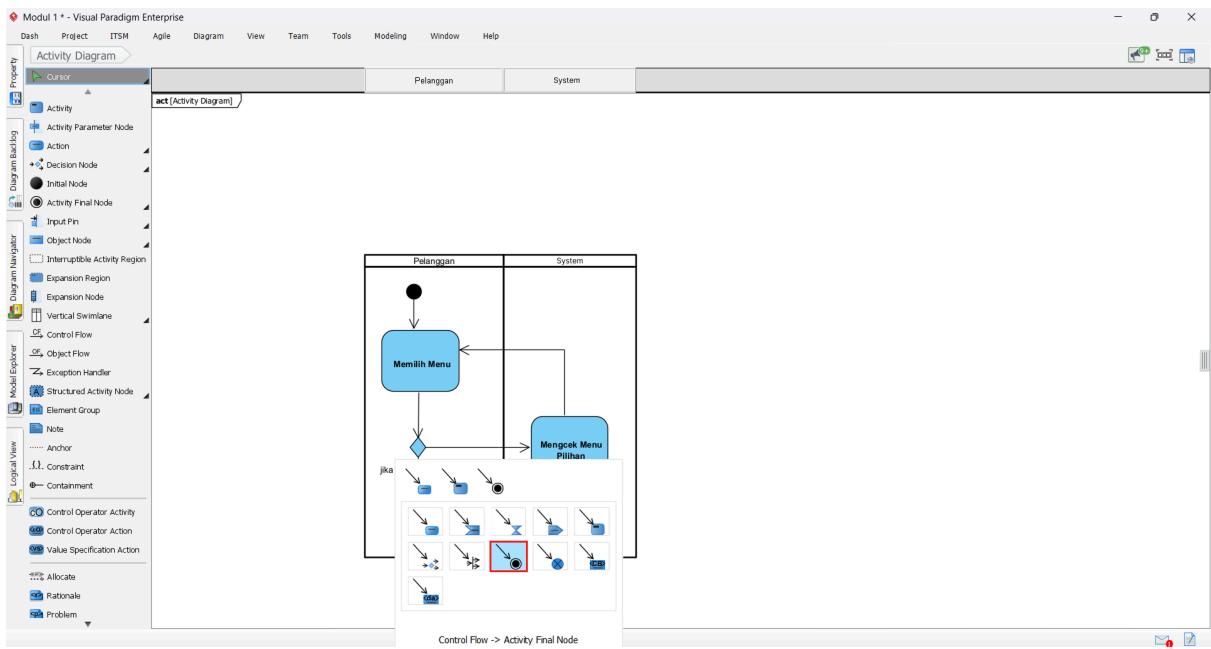




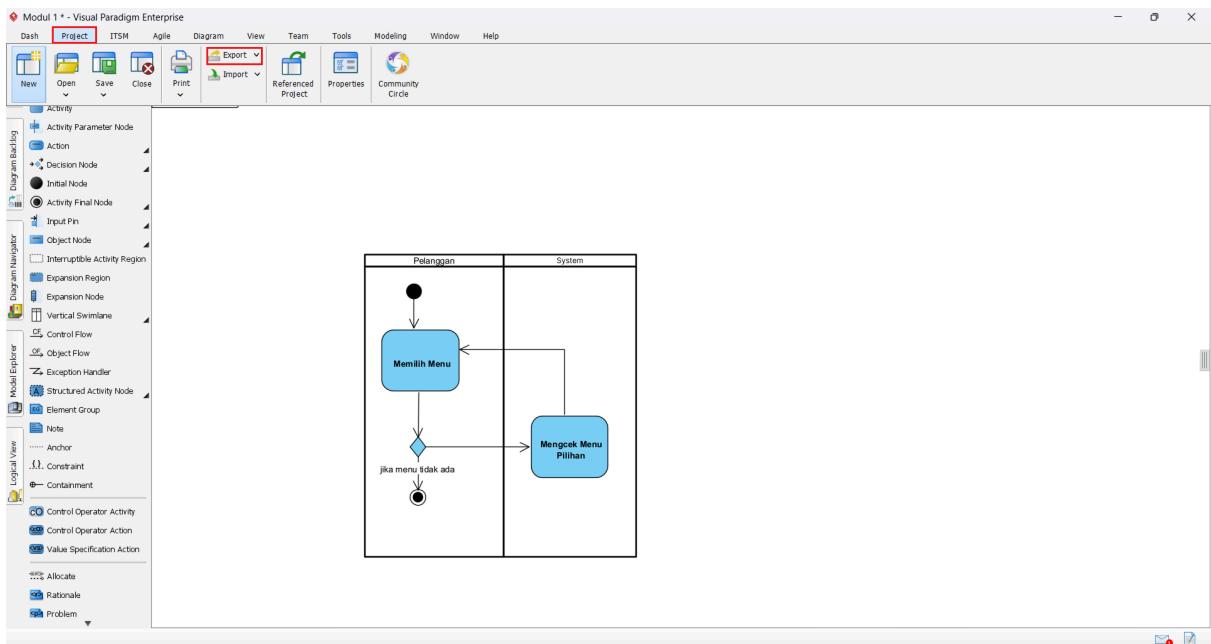
18. klik decision node yang berada di dalam Swimlane, lalu klik Resource Catalog yang muncul di kanan atas notasi tersebut.



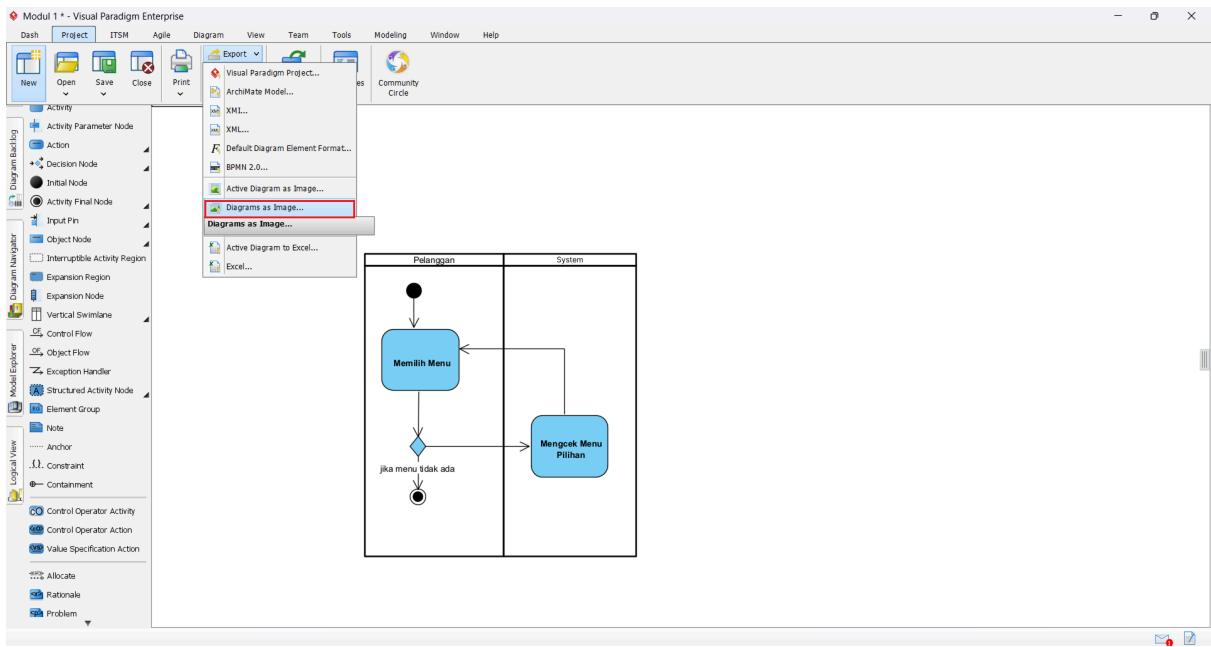
19. Jika Activity Diagram dari suatu fitur sudah selesai, tambahkan Activity Final Node sebagai notasi akhir untuk menandai berhentinya alur pada diagram tersebut.



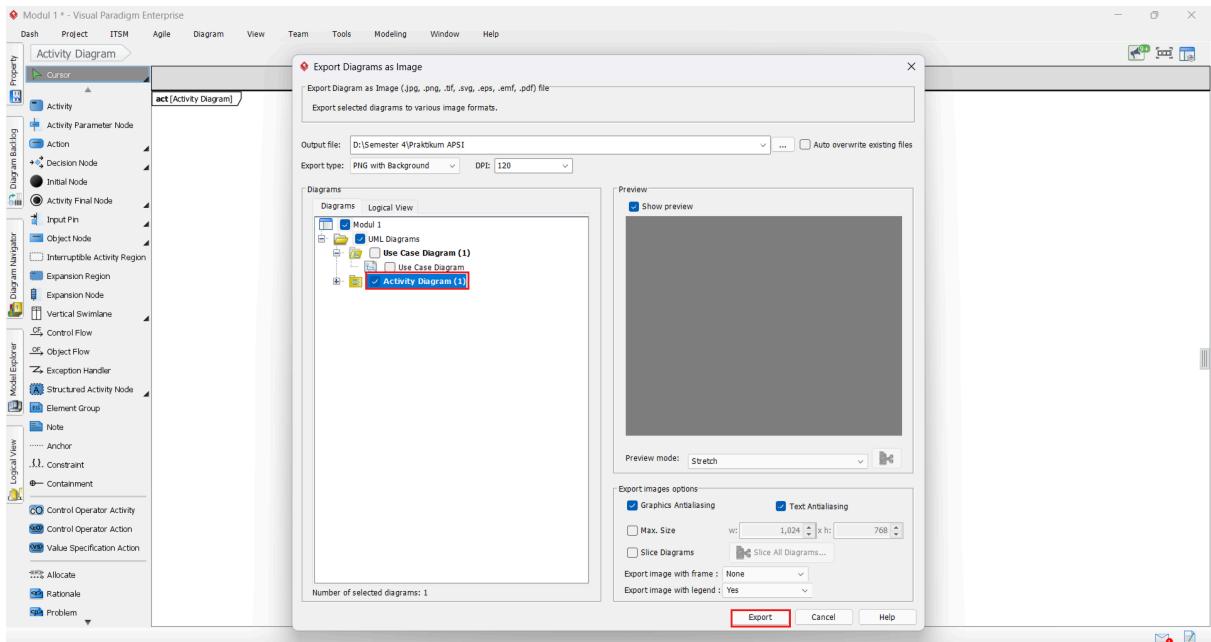
20. Untuk mengeksport Activity Diagram, buka tab Project, lalu klik Export.



21. Setelah itu, pilih opsi Diagram as Image untuk mengekspor diagram dalam format gambar.



22. Lalu, centang diagram yang ingin diekspor, tentukan lokasi penyimpanan file, dan klik tombol Export untuk menyelesaikan proses.



## CONTOH STUDI KASUS

### A. Studi Kasus 1 (Level Mudah)

KostKu merupakan sebuah platform digital yang dikembangkan oleh seorang wirausaha muda yang pernah mengalami kesulitan dalam mencari tempat tinggal ketika menempuh pendidikan di luar kota. Berangkat dari pengalamannya tersebut, ia menciptakan KostKu sebagai solusi untuk membantu mahasiswa dalam menemukan tempat kost yang nyaman, aman, strategis, dan tentunya sesuai dengan kemampuan finansial mereka.

Sejak pertama kali diluncurkan, KostKu berhasil menarik perhatian banyak mahasiswa karena fitur-fiturnya yang sangat membantu. Melalui platform ini, mahasiswa dapat mencari kost berdasarkan lokasi, harga sewa, fasilitas, dan informasi lainnya yang ditampilkan secara lengkap melalui foto dan deskripsi. Dengan begitu, proses pencarian menjadi lebih mudah dan efisien karena mahasiswa tidak perlu lagi datang langsung ke lokasi untuk survei.

Seiring dengan meningkatnya jumlah pengguna setiap harinya, pendiri KostKu menyadari bahwa sistem operasional yang ada perlu dioptimalkan agar lebih terintegrasi dan efisien. Untuk itu, ia mengembangkan sebuah sistem layanan berbasis web yang dinamakan Portal Web KostKu. Sistem ini dirancang khusus untuk dua jenis pengguna utama, yaitu pemilik kost dan mahasiswa.

Dalam portal tersebut, pemilik kost dapat mendaftarkan akun untuk menambahkan informasi mengenai unit kost yang mereka miliki, seperti lokasi, harga sewa, fasilitas, serta jumlah kamar yang tersedia. Jika kamar sudah penuh atau tidak lagi disewakan, pemilik kost juga dapat menghapus atau menonaktifkan unit tersebut dari portal. Sementara itu, mahasiswa dapat menggunakan portal ini untuk mencari kost berdasarkan berbagai kriteria yang relevan. Setelah menemukan kost yang sesuai, mahasiswa dapat melihat detail lengkap dari unit kost dan menghubungi pemilik kost melalui informasi kontak yang tersedia.

Portal Web KostKu dirancang dengan antarmuka yang sederhana, informatif, dan mudah digunakan, sehingga dapat diakses kapan saja dan di mana saja. Diharapkan, platform ini dapat menjadi solusi utama bagi mahasiswa dalam mencari tempat

tinggal yang sesuai dengan kebutuhan mereka selama menjalani masa studi.

## 1. Functional Requirement

- Berikut contoh Functional Requirement pada Portal Web KostKu

NO	Aktor	Deskripsi Aktor	Fungsionalitas Sistem
1	Pemilik Kost	Pengguna yang mengelola informasi dan data unit kost pada portal web KostHub	<ol style="list-style-type: none"><li>1. Pemilik Kost dapat menambahkan unit kost</li><li>2. Pemilik Kost dapat menghapus unit kost</li><li>3. Pemilik Kost dapat mengecek status unit kamar</li></ol>
2	Mahasiswa	Pengguna yang menggunakan layanan pencarian kost melalui Portal KostHub	<ol style="list-style-type: none"><li>1. Mahasiswa dapat mencari unit kost berdasarkan kriteria tertentu</li><li>2. Mahasiswa dapat melihat detail kost</li><li>3. Mahasiswa dapat menghubungi pemilik kost</li></ol>

## 2. Use Case Diagram

- Berikut contoh Use Case Diagram pada Portal Web KostKu



## 3. Use Case Description

- Berikut contoh Use Case Description pada Fitur Menambahkan Kamar atau Unit Kost

Name	Menambahkan Unit Kost
Description	Kegiatan untuk menambahkan informasi kamar kost ke dalam portal web KostHub.
Precondition	Pemilik Kost berada pada halaman "Tambah Kamar Kost".
Postcondition	Sistem menampilkan kamar kost yang berhasil ditambahkan dan tersedia untuk dilihat oleh mahasiswa.
Error Situation	Pemilik Kost gagal menambahkan kamar kost.
System state in the event of an error	Sistem tidak menyimpan data kamar kost dan menampilkan pesan kesalahan.
Actor	Pemilik Kost

Trigger	Pemilik Kost ingin menampilkan kamar kost agar bisa dilihat dan disewa oleh mahasiswa.
Standard in Process	<ol style="list-style-type: none"><li>1. Pemilik Kost masuk ke halaman "Tambah Kamar Kost"</li><li>2. Sistem menampilkan form penambahan kamar kost.</li><li>3. Pemilik Kost mengisi informasi lokasi kost.</li><li>4. Sistem menampilkan hasil input lokasi.</li><li>5. Pemilik Kost mengisi harga sewa bulanan.</li><li>6. Pemilik Kost memilih fasilitas yang tersedia di kamar kost (AC, kamar mandi dalam, WiFi, dll).</li><li>7. Pemilik Kost mengunggah foto kamar kost.</li><li>8. Pemilik Kost menekan tombol "Submit".</li><li>9. Sistem menyimpan data kamar kost.</li><li>10. Sistem menampilkan kamar kost yang telah berhasil ditambahkan.</li></ol>
Alternative Processes	<ol style="list-style-type: none"><li>1. Pemilik Kost masuk ke halaman "Tambah Kamar Kost".</li><li>2. Sistem menampilkan form penambahan kamar kost.</li><li>3. Pemilik Kost mengisi informasi lokasi kost.</li><li>4. Sistem menampilkan hasil input lokasi.</li><li>5. Pemilik Kost mengisi harga sewa bulanan.</li></ol>

	<ol style="list-style-type: none"><li>6. Pemilik Kost memilih fasilitas yang tersedia di kamar kost (AC, kamar mandi dalam, WiFi, dll).</li><li>7. Pemilik Kost mengunggah foto kamar kost.</li><li>8. Pemilik Kost menekan tombol "Submit".</li><li>9. Sistem gagal menyimpan data kamar kost.</li><li>10. Sistem mengusulkan pemilik kost untuk mencoba kembali atau mengecek koneksi internet.</li><li>11. Pemilik Kost mencoba kembali.</li><li>12. Pemilik Kost berhasil menambahkan kamar kost setelah mencoba kembali.</li></ol>
--	---

- Berikut contoh Use Case Description pada Fitur Menghapus Kamar atau Unit Kost

Name	Menghapus Unit Kost
Description	Proses penghapusan data kamar oleh dari portal web KostHub.
Precondition	Pemilik Kost telah login dan berada pada halaman "Hapus Unit Kost".
Postcondition	Sistem berhasil menghapus data kamar kost dari daftar yang tersedia.
Error Situation	Pemilik Kost tidak dapat menghapus Unit kamar kost.
System state in the event of an error	Data kamar kost tidak berubah, dan sistem menampilkan pesan kesalahan.
Actor	Pemilik Kost

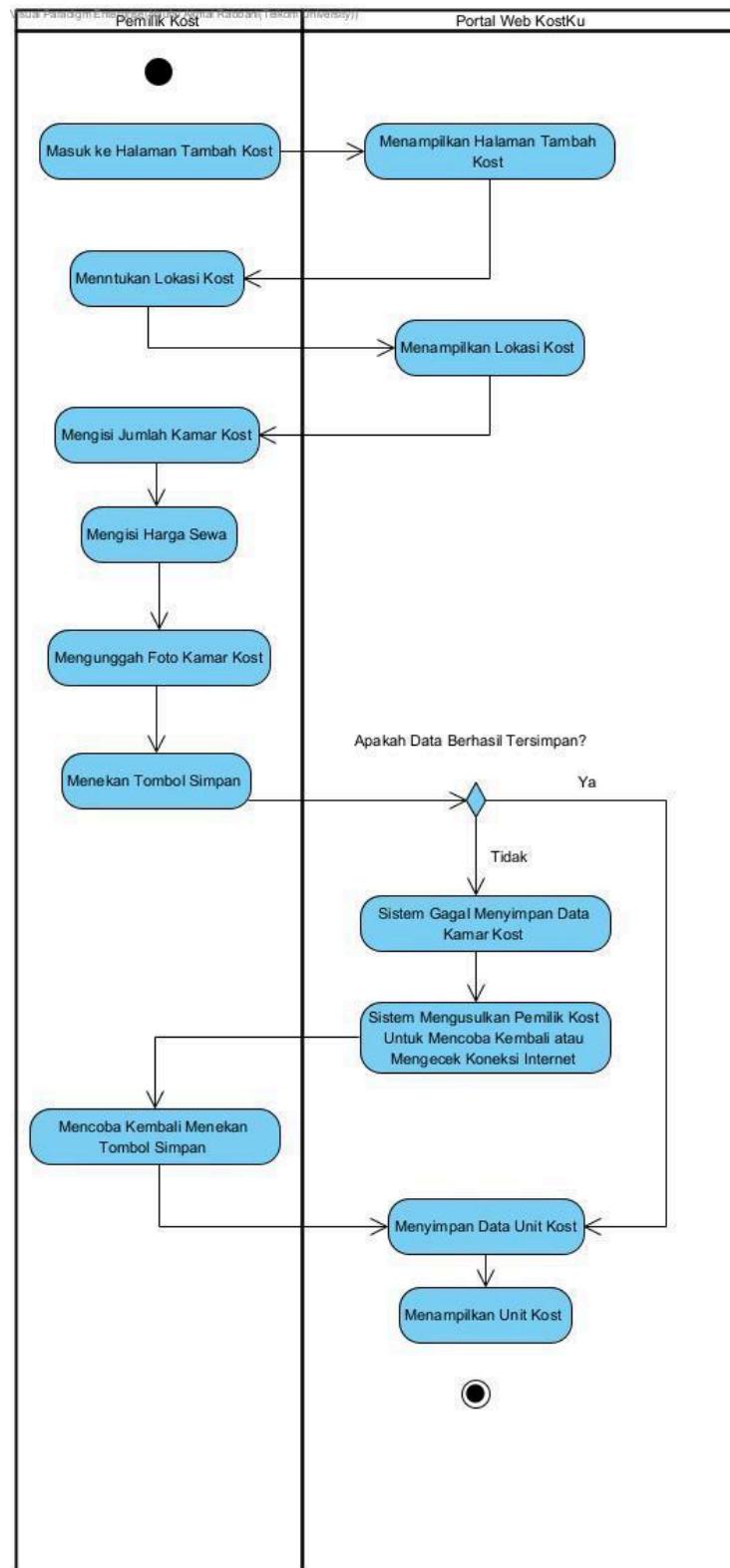
Trigger	Pemilik Kost ingin menghapus kamar kost yang sudah tidak tersedia/disewakan.
Standard in Process	<ol style="list-style-type: none"><li>1. Pemilik Kost masuk ke halaman "Kelola Kamar Kost".</li><li>2. Sistem menampilkan daftar kamar kost yang telah ditambahkan.</li><li>3. Pemilik Kost memilih kamar kost yang ingin dihapus.</li><li>4. Sistem menampilkan detail kamar kost yang dipilih.</li><li>5. Pemilik Kost menekan tombol "Hapus".</li><li>6. Sistem meminta konfirmasi penghapusan.</li><li>7. Pemilik Kost menekan tombol konfirmasi "Ya".</li><li>8. Sistem menghapus kamar kost dari database dan memperbarui daftar.</li></ol>
Alternative Processes	<ol style="list-style-type: none"><li>1. Pemilik Kost masuk ke halaman "Tambah Kamar Kost".</li><li>2. Sistem menampilkan form penambahan kamar kost.</li><li>3. Pemilik Kost mengisi informasi lokasi kost.</li><li>4. Sistem menampilkan hasil input lokasi.</li><li>5. Pemilik Kost mengisi harga sewa bulanan.</li><li>6. Pemilik Kost memilih fasilitas yang tersedia di kamar kost (AC, kamar mandi dalam, WiFi, dll).</li><li>7. Pemilik Kost mengunggah foto kamar kost.</li><li>8. Pemilik Kost menekan tombol "Submit".</li><li>9. Sistem gagal menyimpan data kamar kost.</li><li>10. Sistem mengusulkan pemilik kost untuk mencoba kembali atau mengecek koneksi internet.</li></ol>

- |  |   |
|--|---|
|  | <ol style="list-style-type: none"><li>11. Pemilik Kost mencoba kembali.</li><li>12. Pemilik Kost berhasil menambahkan kamar kost setelah mencoba kembali.</li></ol> |
|--|---|

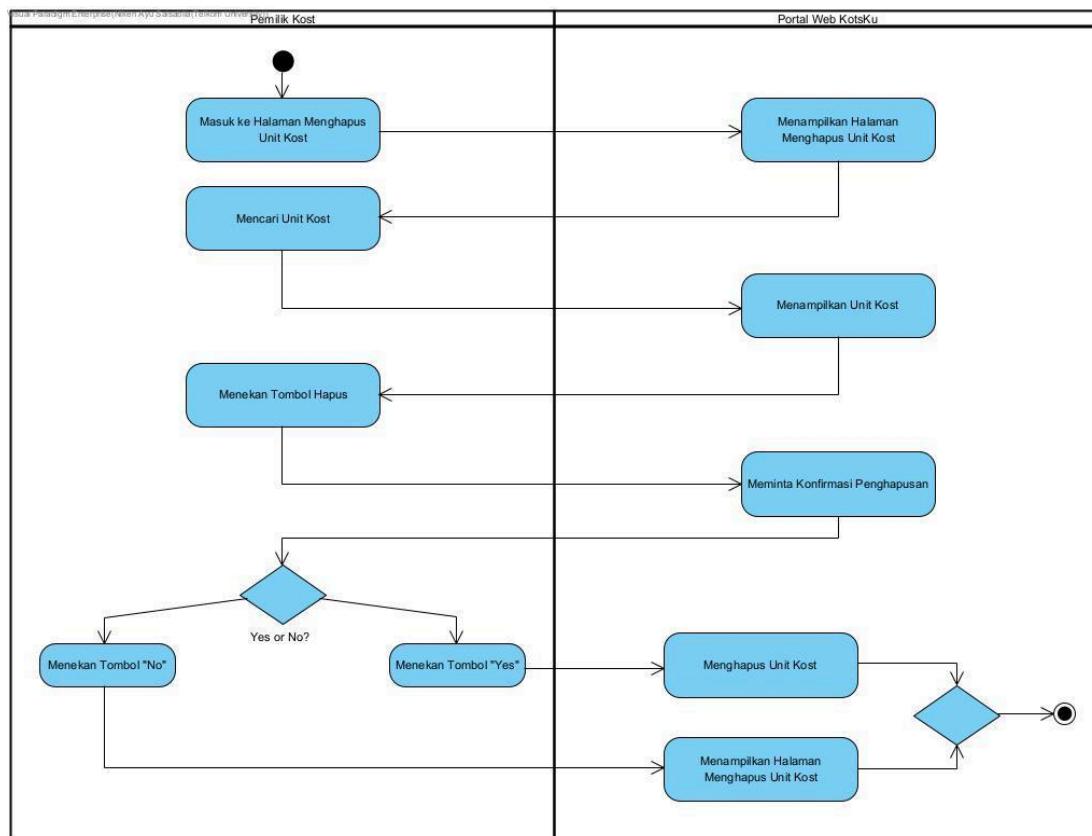
**(Dibuat dengan langkah yang sama untuk fitur lainnya)**

#### 4. Activity Diagram

- Berikut contoh Activity Diagram pada Fitur Menambahkan Kamar atau Unit Kost



- Berikut contoh Activity Diagram pada Fitur Menghapus Kamar atau Unit Kost



**(Dibuat dengan langkah yang sama untuk fitur lainnya)**

## B. Studi Kasus 1 (Level Mudah)

BelajarIn merupakan sebuah platform layanan kursus online yang berbasis di kota Bandung. Platform ini dirancang untuk membantu mahasiswa dan masyarakat umum dalam meningkatkan keterampilan di berbagai bidang, seperti pemrograman, desain grafis, bahasa asing, hingga digital marketing. BelajarIn didirikan oleh sekelompok alumni Telkom University yang peduli terhadap peningkatan kompetensi anak muda di era digital. Seiring dengan meningkatnya minat masyarakat terhadap pembelajaran secara daring, pihak pengelola BelajarIn terus berupaya untuk meningkatkan mutu layanan serta memperluas jangkauan pengguna. BelajarIn tidak hanya menawarkan kelas secara daring, tetapi juga menyediakan fitur interaktif seperti forum diskusi, penilaian otomatis, serta layanan konsultasi jadwal dengan mentor secara online.

BelajarIn memiliki visi menjadi platform kursus online yang terpercaya, mudah diakses, dan berkualitas tinggi dalam mendukung pengembangan keterampilan masyarakat Indonesia. Untuk mewujudkan visi tersebut, pihak pengelola membangun sebuah sistem portal website terintegrasi bernama Portal Web EduCourseHub. Portal web ini akan menjadi pusat dari seluruh proses layanan digital yang dapat diakses oleh empat jenis pengguna, yaitu Admin Platform, Mentor, Peserta Baru, dan Peserta Aktif.

Sebelum dapat mengakses layanan kursus, Peserta Baru diwajibkan untuk melakukan registrasi akun pada Portal Web BelajarIn. Setelah berhasil melakukan registrasi, sistem akan mengarahkan peserta untuk mengisi data diri secara lengkap. Selanjutnya, Peserta Baru dapat mengakses katalog kursus dan memilih kelas yang ingin diikuti. Setelah memilih kursus, peserta akan diarahkan ke halaman informasi jadwal dan biaya kursus, kemudian menuju ke halaman pembayaran. Jika proses pembayaran telah diverifikasi oleh Admin Platform, peserta akan mendapatkan akses ke materi kursus dan dapat langsung menjadwalkan sesi konsultasi dengan mentor yang tersedia.

Sementara itu, bagi Peserta Aktif yang sebelumnya telah terdaftar dan mengikuti kursus di Portal Web BelajarIn, mereka dapat langsung login dan mengakses kursus yang sedang berjalan.

Peserta Aktif dapat melihat materi, bergabung di forum diskusi, serta menjadwalkan sesi konsultasi jika mengalami kendala dalam proses belajar. Jika ingin mengikuti kursus tambahan, Peserta Aktif dapat kembali memilih kursus baru dan melakukan proses pendaftaran seperti halnya Peserta Baru. BelajarIn juga menyediakan fitur pembaruan data diri jika terdapat perubahan informasi seperti email, kontak, atau domisili peserta.

Mentor pada Portal Web BelajarIn memiliki akses untuk melihat jadwal konsultasi yang telah ditentukan oleh peserta, memberikan umpan balik atau penilaian terhadap perkembangan belajar, serta memperbarui materi kursus. Mentor juga dapat berinteraksi langsung dengan peserta melalui forum diskusi atau sesi konsultasi daring yang dijadwalkan.

Admin Platform memiliki peran penting dalam mengelola sistem secara keseluruhan. Admin bertugas melakukan verifikasi pembayaran peserta, menangani bentrokan jadwal antara peserta dan mentor, serta memfasilitasi pembaruan kursus atau data mentor. Admin juga berwenang dalam mengelola hak akses dari setiap jenis pengguna, menjaga keamanan sistem, dan memastikan seluruh layanan berjalan dengan baik.

Dengan pengembangan Portal Web BelajarIn ini, diharapkan layanan kursus online dapat berjalan lebih efektif, efisien, serta mampu memberikan pengalaman belajar yang berkualitas bagi seluruh penggunanya.

## 1. Functional Requirement

- Berikut contoh Functional Requirement pada Portal Web BelajarIn

	Aktor	Deskripsi Aktor	Fungsionalitas Sistem
1	Peserta Baru	User yang baru mendaftar untuk mengikuti kursus	<ol style="list-style-type: none"><li>1. Peserta Baru dapat melakukan registrasi pada portal web</li></ol>
			<ol style="list-style-type: none"><li>2. Peserta Baru dapat melakukan login menggunakan akun yang telah terdaftar</li></ol>

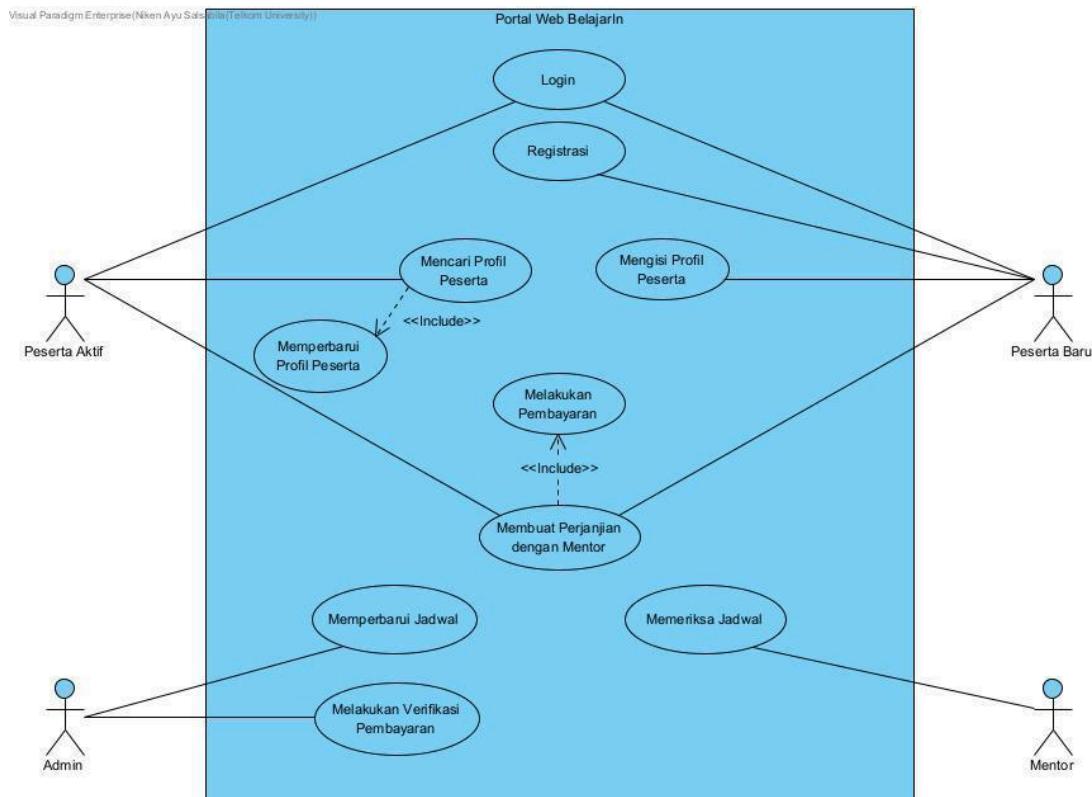
			<ul style="list-style-type: none"><li>3. Peserta Baru dapat mengisi data diri secara lengkap</li><li>4. Peserta Baru dapat memilih dan mendaftar untuk kursus yang diinginkan</li><li>5. Peserta Baru dapat melihat jadwal dan biaya kursus</li><li>6. Peserta Baru dapat melakukan pembayaran untuk kursus yang dipilih</li></ul>
2	Peserta Aktif	User yang sudah terdaftar dan sedang mengikuti kursus	<ul style="list-style-type: none"><li>1. Peserta Aktif dapat melakukan login</li><li>2. Peserta Aktif dapat melihat status kursus yang sedang diikuti</li><li>3. Peserta Aktif dapat mengakses materi kursus yang telah disediakan</li><li>4. Peserta Aktif dapat berinteraksi dengan mentor melalui forum diskusi</li><li>5. Peserta Aktif dapat melakukan penjadwalan konsultasi dengan mentor</li><li>6. Peserta Aktif dapat memperbarui data profil jika diperlukan</li></ul>
3	Mentor	User yang bertanggung jawab untuk mengajar dan membimbing peserta	<ul style="list-style-type: none"><li>1. Mentor dapat login untuk mengakses portal</li><li>2. Mentor dapat</li></ul>

			<p>melihat daftar peserta yang terdaftar di kursus yang mereka ajar</p>
			<p>3. Mentor dapat memperbarui materi kursus yang sedang diajarkan</p>
			<p>4. Mentor dapat memberikan umpan balik atau penilaian kepada peserta</p>
			<p>5. Mentor dapat mengatur dan melihat jadwal konsultasi dengan peserta</p>
4	Admin	User yang bertugas untuk mengelola sistem dan pengguna	<p>1. Admin dapat melakukan login untuk mengakses panel administrasi</p> <p>2. Admin dapat menambah, mengedit, atau menghapus kursus</p> <p>3. Admin dapat mengelola pengguna, termasuk menonaktifkan atau mengaktifkan akun</p> <p>4. Admin dapat melakukan verifikasi pembayaran peserta</p> <p>5. Admin dapat melihat laporan penggunaan dan statistik kursus</p> <p>6. Admin dapat memperbarui jadwal kelas dan konsultasi jika terjadi bentrok</p>

			7. Admin dapat mengelola hak akses untuk mentor dan peserta
--	--	--	---

## 2. Use Case Diagram

- Berikut contoh Use Case Diagram pada Portal Web BelajarIn



## 3. Use Case Description

- Berikut contoh Use Case Description pada Fitur Login

Name	Melakukan Login
Description	Kegiatan login Peserta untuk masuk ke akun di Portal Web BelajarIn
Precondition	Peserta sudah berada pada halaman Login
Postcondition	Sistem menampilkan halaman dashboard BelajarIn sesuai role

	<b>peserta</b>
Error Situation	Peserta tidak dapat melakukan login
System state in the event of an error	Peserta belum berhasil masuk ke sistem karena email atau password salah
Actor	Admin, Peserta Baru, Peserta Aktif, Mentor
Trigger	Peserta perlu login untuk mengakses fitur pada Portal Web BelajarIn
Standard in Process	<ol style="list-style-type: none"> <li>1. Peserta masuk ke halaman login</li> <li>2. Sistem menampilkan halaman login</li> <li>3. Peserta memasukkan email &amp; password</li> <li>4. Peserta menekan tombol login</li> <li>5. Sistem mengecek email &amp; password</li> <li>6. Jika sesuai, sistem menampilkan konfirmasi login berhasil</li> <li>7. Sistem mengarahkan ke halaman dashboard sesuai peran Peserta</li> </ol>
Alternative Processes	<p>1' Peserta masuk ke halaman login</p> <p>2' Sistem menampilkan halaman login</p> <p>3' Peserta memasukkan email &amp; password</p>

	4' Peserta menekan tombol login  5' Sistem mengecek email & password  6' Jika salah, sistem menampilkan notifikasi error  7' Peserta menginput ulang email & password  8' Peserta menekan tombol login  9' Sistem menampilkan konfirmasi login berhasil  10' Sistem menampilkan halaman dashboard
--	---

- Berikut contoh Use Case Description pada Fitur Mengelola Jadwal Kelas

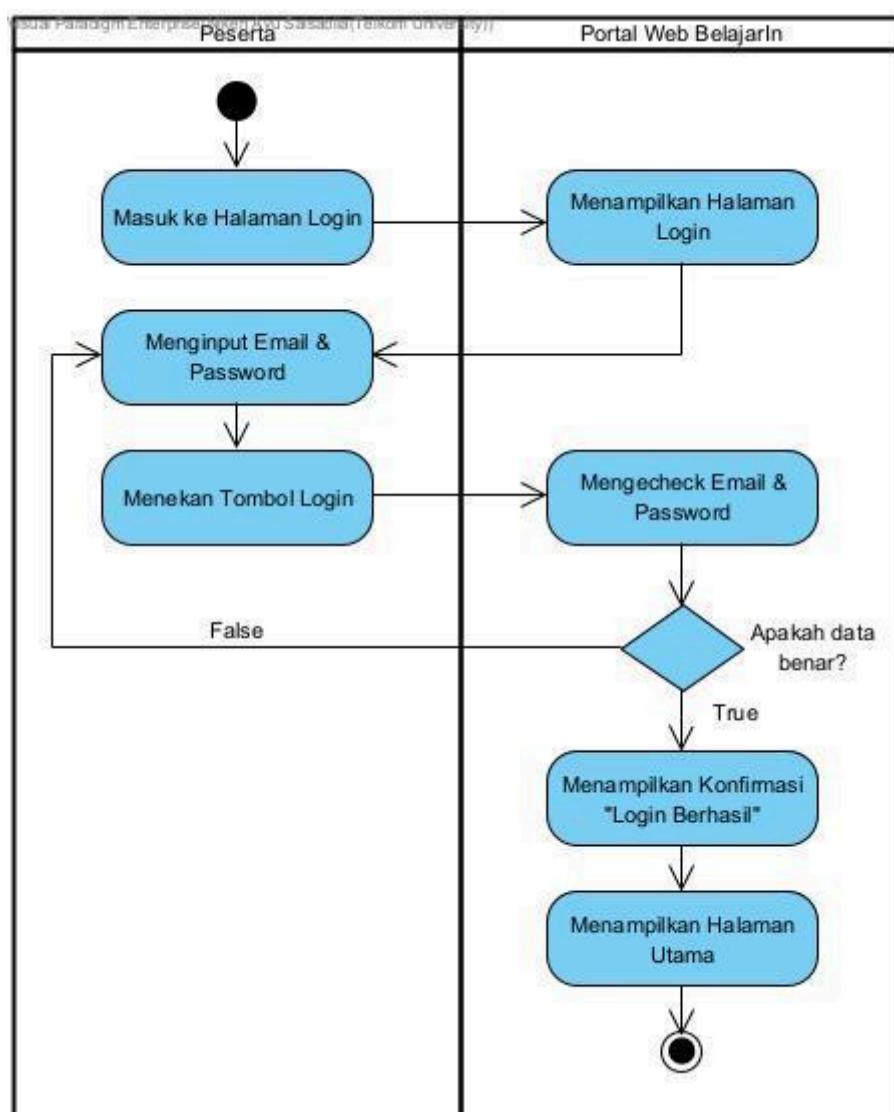
Name	Mengelola Jadwal Kelas
Description	Kegiatan mengatur atau memperbarui jadwal kelas online di BelajarIn
Precondition	Admin ingin menambahkan atau mengubah jadwal kelas
Postcondition	Sistem menampilkan jadwal kelas terbaru di portal
Error Situation	Admin tidak dapat memperbarui jadwal kelas
System state in the event of an error	Admin belum mengisi data jadwal kelas dengan lengkap
Actor	Admin
Trigger	Admin perlu mengatur jadwal kelas baru atau memperbarui jadwal lama
Standard in Process	1. Admin masuk ke halaman jadwal kelas

	<ol style="list-style-type: none"><li>2. Sistem menampilkan daftar jadwal yang ada</li><li>3. Admin memilih jadwal yang ingin diubah atau membuat jadwal baru</li><li>4. Sistem menampilkan form pengisian atau pengeditan jadwal</li><li>5. Admin mengisi data (nama kelas, waktu, instruktur)</li><li>6. Admin menekan tombol submit</li><li>7. Sistem menampilkan konfirmasi jadwal berhasil diperbarui</li><li>8. Sistem memperbarui tampilan daftar jadwal</li></ol>
Alternative Processes	<ol style="list-style-type: none"><li>1' Admin masuk ke halaman jadwal kelas</li><li>2' Sistem menampilkan daftar jadwal</li><li>3' Admin memilih jadwal untuk diperbarui</li><li>4' Sistem menampilkan form pengeditan</li><li>5' Admin salah input data jadwal</li><li>6' Admin menekan tombol "Batal"</li><li>7' Sistem menampilkan kembali form kosong</li><li>8' Admin mengisi ulang data dengan benar</li></ol>

	9' Admin menekan tombol submit  10' Sistem menampilkan konfirmasi jadwal berhasil diperbarui  11' Sistem memperbarui daftar jadwal terbaru
--	--

#### 4. Activity Diagram

- Berikut contoh Activity Diagram pada Fitur Login



## DAFTAR PUSTAKA

Dennis, A., Wixom, B. H., & Tegarden, D. (2015). System Analysis & Design (An Object-Oriented Approach with UML) - Fifth Edition. Wiley.

Seidl, M., Huemer, C., Scholz, M., & Kappel, G. (2015). UML @ Classroom (An Introduction to Object-Oriented Modeling). Springer.

10.1007/978-3-319-12742-2