



Analisis Perancangan Sistem Informasi

Tahun 2025

Modul 2

Class Diagram & Sequence Diagram

Arranged By:
EAD Laboratory Team

Asisten Praktikum



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
Aswangga P.



FRHN
Muhammad Ilyas



ALEM
Matthew Alexander



BGND
Stefanus Jesano P.



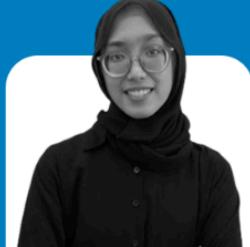
YARE
Muhammad Ikhtiar



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

DAFTAR ISI	2
PERATURAN PRAKTIKUM	3
A. Class Diagram	7
1. Manfaat Class Diagram dalam Pengembangan	7
2. Komponen Utama Class Diagram	8
3. Hubungan Antar Kelas	9
B. Sequence Diagram	12
1. Definisi	12
2. Notasi Sequence Diagram	13
3. Entity-Controller-Boundary (ECB)	22
4. Pedoman Membuat Sequence Diagram	23
LANGKAH-LANGKAH PRAKTIKUM	24
A. Cara Membuat Class Diagram	24
B. Cara Membuat Sequence Diagram	33
CONTOH	48
A. Studi Kasus 1 (Level Mudah)	48
B. Studi Kasus (Level Kompleks)	52

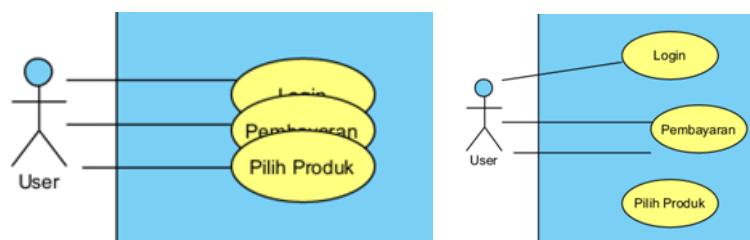
PERATURAN PRAKTIKUM

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 plottingan, 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.

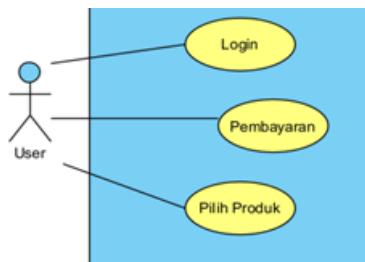
4. Peraturan Penggeraan

- 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. Penggeraan 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 penggeraan berupa file format PDF.
- i. Format Pengumpulan Hasil Penggeraan:
 - File penamaan Jurnal dalam bentuk PDF:

APSI_MODULX_KODEASISTEN_NAMA LENGKAP_NIM

Contoh:

APSI_MODUL1_JHON_ARI ANANTA_1021202200000

- File penamaan Tugas Pendahuluan dalam bentuk PDF:

APSI_TPX_KODEASISTEN_NAMA LENGKAP_NIM

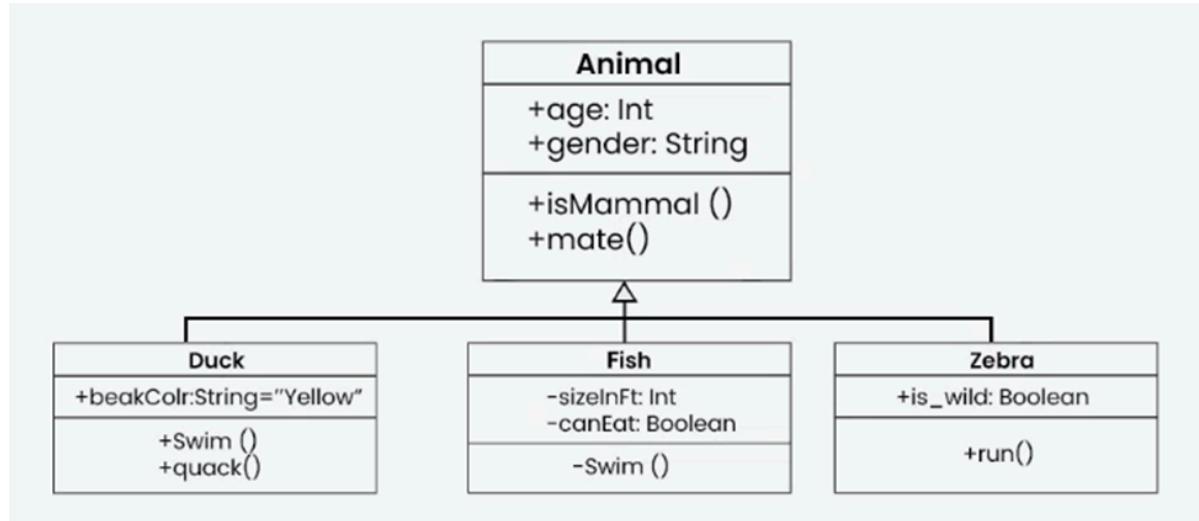
Contoh:

APSI_TP1_JHON_ARI ANANTA_1021202200000

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

LANDASAN TEORI

A. Class Diagram



Class diagram adalah alat visual dalam UML yang digunakan untuk menggambarkan struktur statis dari sebuah sistem perangkat lunak. Struktur statis berarti diagram ini tidak fokus pada alur atau proses (yang biasanya ditangani oleh diagram lain seperti sequence diagram), melainkan pada "apa" yang ada dalam sistem, yaitu kelas-kelas beserta hubungannya. Dalam paradigma berorientasi objek, kelas adalah inti dari sistem karena kelas menjadi dasar untuk membuat objek—unit nyata yang akan berjalan dalam program. Dengan class diagram, pengembang dapat merancang sistem sebelum menulis kode, sehingga mengurangi kesalahan dan mempercepat proses pengembangan.

1. Manfaat Class Diagram dalam Pengembangan

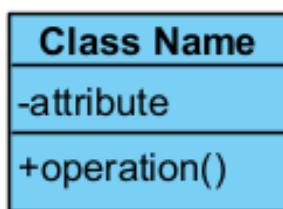
Class diagram memberikan gambaran tingkat tinggi (high-level overview) tentang sistem, sehingga:

- Mempermudah Komunikasi: Tim pengembang, analis, dan klien dapat memahami desain tanpa perlu melihat kode langsung.
- Dokumentasi: Diagram ini menjadi referensi yang berguna selama dan setelah pengembangan selesai.
- Desain Berorientasi Objek: Class diagram mendukung prinsip-prinsip seperti enkapsulasi (menyembunyikan detail), pewarisan (inheritance), dan polimorfisme (perilaku berbeda untuk kelas turunan).

Dalam siklus pengembangan perangkat lunak, class diagram biasanya dibuat pada tahap analisis dan desain, lalu digunakan sebagai panduan saat menulis kode pada tahap implementasi. Diagram ini juga membantu mendeteksi masalah desain sejak dini, seperti hubungan yang tidak logis atau kelas yang terlalu kompleks.

2. Komponen Utama Class Diagram

Class diagram terdiri dari beberapa elemen kunci yang membentuk struktur dasarnya. Berikut penjelasan masing-masing komponen:



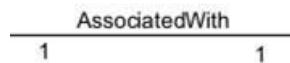
1. Kelas (Class)
 - a. Definisi: Kelas adalah cetak biru (blueprint) untuk membuat objek. Kelas mendefinisikan data (atribut) dan perilaku (metode) yang dimiliki oleh objek yang dihasilkan darinya.
 - b. Representasi: Dalam diagram, kelas digambarkan sebagai kotak persegi panjang yang terbagi menjadi tiga bagian:
 - Bagian atas: Nama kelas (biasanya diawali huruf kapital).
 - Bagian tengah: Daftar atribut.
 - Bagian bawah: Daftar metode.
2. Atribut (Attributes)
 - a. Definisi: Atribut adalah variabel yang menyimpan data atau keadaan dari kelas. Atribut menjelaskan properti yang dimiliki oleh objek.
 - b. Format: Ditulis dengan visibilitas namaAtribut: tipeData.
 - c. Visibilitas:
 - o + (public): Dapat diakses dari luar kelas.
 - o - (private): Hanya dapat diakses dari dalam kelas.
 - o # (protected): Dapat diakses dari dalam kelas dan kelas turunannya.
- Contoh: `- warna: String` menunjukkan atribut "warna" bertipe String dan bersifat private.
3. Methods

Method atau operation adalah fungsi atau prosedur yang dimiliki oleh suatu kelas untuk melakukan suatu tindakan atau operasi. Methods biasanya didefinisikan di dalam kelas dan dapat dipanggil oleh objek dari kelas tersebut.

3. Hubungan Antar Kelas

Hubungan antar kelas menunjukkan bagaimana kelas-kelas saling berinteraksi atau terhubung. Berikut adalah jenis-jenis hubungan utama dalam class diagram:

1. Asosiasi (Association)



- a. Definisi: Hubungan ini mewakili hubungan umum antar class. Hubungan ini bisa menunjukkan interaksi antar objek dari class yang berbeda. Hubungan asosiasi biasanya dilengkapi dengan simbol multiplikasi yang menunjukkan berapa banyak objek dari satu class dapat berhubungan dengan objek dari class lainnya. Berikut adalah beberapa contoh multiplikasi:

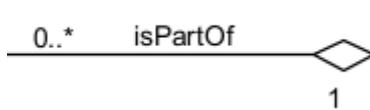
Nilai Kardinalitas	Arti	Contoh	
0..1	Nol atau satu	karyawan	0..1 istri
1	Hanya satu	negara	1 presiden
0..*	Nol atau lebih	karyawan	0..* anak
1..*	Satu atau lebih	bos	1..* bawahan
n	Hanya n (dengan n > 1)	karyawan	n cek up
0..n	Nol sampai n (dengan n > 1)	karyawan	0..n sim
1..n	Satu sampai n (dengan n > 1)	kereta api	1..n gerbong

- b. Representasi: Garis lurus antara dua kelas.
c. Contoh:



Sebagai contoh, class "Pengemudi" dapat memiliki hubungan asosiasi dengan class "Mobil" yang menunjukkan bahwa seorang Pengemudi dapat menyetir sebuah Mobil.

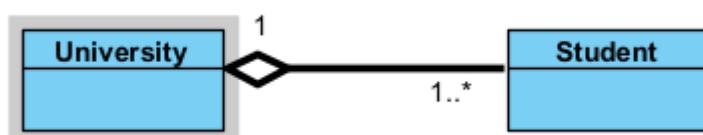
2. Agregasi (Aggregation)



- a. Definisi: Agregasi adalah hubungan antara dua kelas di mana satu kelas merupakan bagian dari kelas lainnya, tetapi tetap bisa berdiri sendiri. Dengan kata lain, meskipun suatu objek menjadi bagian dari objek lain, ia tetap bisa eksis secara mandiri tanpa bergantung sepenuhnya pada objek tersebut.

Dalam class diagram, agregasi digambarkan dengan garis yang memiliki simbol berlian kosong (\diamond) di sisi kelas yang lebih besar atau yang memiliki bagian.

- b. Representasi: Garis dengan diamond kosong di sisi kelas yang lebih besar atau memiliki bagian
- c. Contoh:

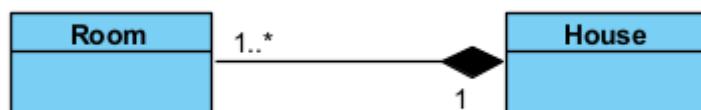


Bayangkan kelas Universitas dan kelas Mahasiswa. Universitas memiliki banyak Mahasiswa, tetapi Mahasiswa tetap bisa ada meskipun tidak terdaftar di Universitas tertentu. Ini adalah contoh agregasi karena Mahasiswa tetap bisa berdiri sendiri meskipun menjadi bagian dari Universitas.

3. Komposisi (Composition)



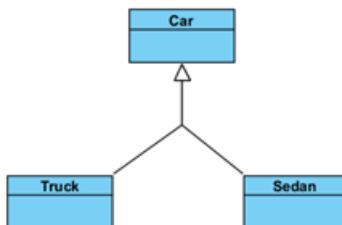
- a. Definisi: Komposisi adalah hubungan yang erat antara dua kelas, di mana satu kelas tidak bisa ada tanpa yang lainnya. Jika objek keseluruhan dihapus, maka bagian-bagiannya juga ikut hilang. Dalam class diagram, komposisi digambarkan dengan garis yang memiliki simbol berlian penuh (\blacklozenge) di sisi kelas yang lebih besar atau yang memiliki bagian.
- b. Representasi: Garis dengan diamond terisi di sisi kelas yang memiliki.
- c. Contoh:



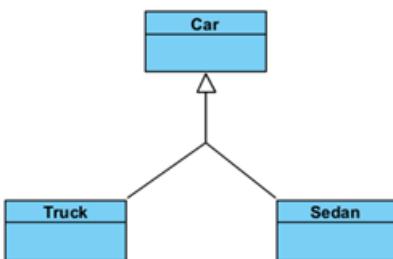
Bayangkan kelas House dan kelas Room. Rumah terdiri dari beberapa ruangan, dan jika rumah dihancurkan, maka

ruangan-ruangan di dalamnya juga ikut hilang. Ruangan tidak bisa berdiri sendiri tanpa Rumah, sehingga ini adalah contoh komposisi.

4. Generalisasi (Generalization)



- Definisi: Generalisasi adalah hubungan antara kelas induk dan kelas anak, yang menunjukkan bahwa suatu kelas adalah versi khusus dari kelas lainnya. Hubungan ini disebut juga sebagai "is-a relationship", yang berarti bahwa kelas anak mewarisi sifat dan perilaku dari kelas induknya. Dalam class diagram, generalisasi digambarkan dengan garis lurus yang memiliki panah kosong (Δ) yang menunjuk ke kelas induk.
- Representasi: Garis dengan panah segitiga kosong menuju kelas induk.
- Contoh:



Car adalah kelas induk, sementara Sedan dan Truck adalah kelas anaknya. Sedan dan Truck mewarisi atribut dan metode dari Car, seperti kecepatan, warna, dan metode untuk mengemudi.

B. Sequence Diagram

1. Definisi

Diagram sequence adalah salah satu jenis diagram dalam UML (*Unified Modelling Language*) yang menjelaskan bagaimana objek-objek dalam sebuah sistem berinteraksi dan berkomunikasi satu sama lain. Bayangkan seperti sebuah alur cerita yang menunjukkan pertukaran pesan antar komponen sistem dari waktu ke waktu.

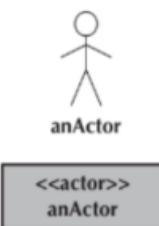
Diagram ini memiliki dua pendekatan utama yang berbeda. Diagram generik menggambarkan berbagai kemungkinan skenario dalam satu use case, memberikan gambaran menyeluruh tentang potensi interaksi dalam sistem dan berguna untuk memahami keseluruhan alur kerja yang mungkin terjadi. Sementara itu, diagram individual fokus pada satu skenario spesifik tertentu,

mendetailkan urutan tepat dari interaksi antar objek dan membantu memahami alur kerja yang sangat konkret.

Diagram sequence paling sering digunakan pada dua tahap penting pengembangan sistem, yaitu tahap analisis dan desain. Pada tahap analisis, diagram ini membantu memahami kebutuhan dan alur sistem, sedangkan pada tahap desain, diagram digunakan untuk merancang detail interaksi antar komponen sistem. Tujuan utamanya adalah menjelaskan dengan jelas bagaimana objek-objek dalam sistem saling berkomunikasi dan bertukar pesan sesuai urutan waktu.

2. Notasi Sequence Diagram

1. Aktor



Aktor adalah individu atau sistem di luar suatu sistem yang dapat mendapatkan manfaat darinya. Mereka berinteraksi dengan sistem dengan mengirim atau menerima pesan.

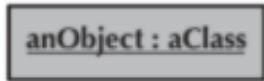
Contoh :



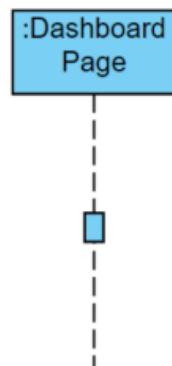
Keterangan:

Dalam contoh yang diberikan, aktor yang berperan dalam use case adalah *Customer*. Selain itu, aktor lain yang dapat terlibat antara lain Mahasiswa, Dosen, dan sebagainya.

2. Object, Lifeline, dan Execution occurrence

Nama	Notasi	Deskripsi
Object		Objek terlibat dalam kolaborasi dengan sistem dengan cara mengirim atau menerima pesan. Biasanya, objek ditempatkan di bagian atas diagram.
Lifeline		Lifeline berfungsi untuk menunjukkan alur Perkembangan dari sebuah objek dalam sebuah urutan.
Execution occurrence		Execution occurrence adalah persegi panjang yang biasanya ditempatkan di atas lifeline. Fungsinya untuk menunjukkan bahwa suatu objek sedang dieksekusi dan berinteraksi dengan objek lain dalam sistem.

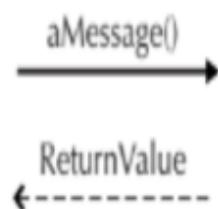
Contoh:



Keterangan:

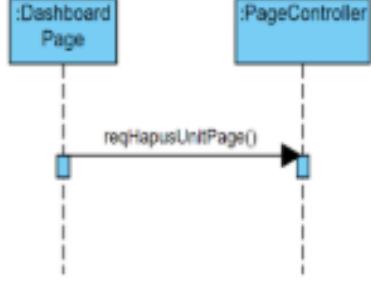
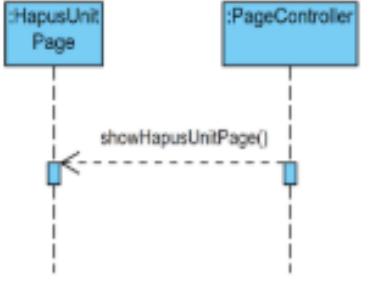
Dalam contoh yang diberikan, terdapat ilustrasi sebuah objek, lifeline, dan execution occurrence yang berperan sebagai boundary, yaitu *DashboardPage*. Selain itu, contoh lain bisa berupa entity atau controller, seperti *PageController*, *UnitApartemen*, dan lain sebagainya.

3. Message



Message digunakan untuk mengirim informasi dari satu objek ke objek lainnya. Pengiriman pesan ditandai dengan panah penuh yang disebut notasi *call message*, sedangkan penerimaan pesan ditandai dengan panah putus-putus yang disebut *reply message*.

Contoh:

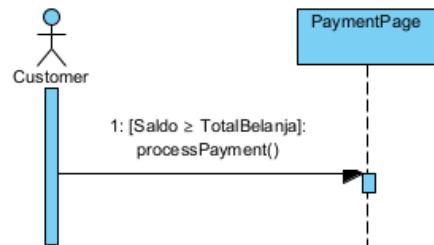
Nama	Contoh	Keterangan
Call Message		Dalam contoh yang diberikan, terdapat ilustrasi sebuah pesan (message). Pada gambar pertama, method <code>reqHapusUnitPage()</code> dikirim dari <code>:DashboardPage</code> ke <code>:PageController</code> .
Reply Message		Sementara itu, pada gambar kedua, method <code>showHapusUnitPage()</code> dikirim sebagai pesan dari controller <code>:PageController</code> ke boundary <code>:HapusUnitPage</code> .

4. Guard Condition



Guard condition adalah syarat yang harus terpenuhi agar suatu pesan bisa dikirim.

Contoh:



Keterangan:

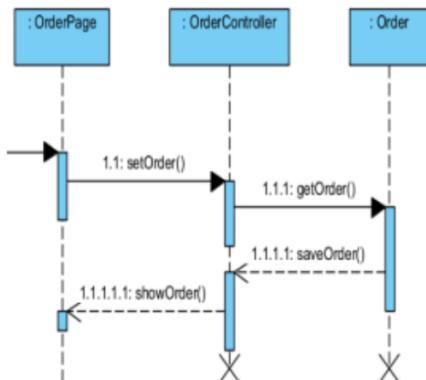
Dalam contoh ini, *Customer* mengirimkan permintaan pembayaran ke *PaymentPage*. Namun, transaksi hanya akan diproses jika syarat $\text{Saldo} \geq \text{TotalBelanja}$ terpenuhi. Jika saldo tidak mencukupi, sistem tidak akan melanjutkan pembayaran.

5. Object Destruction



Object destruction ditandai dengan simbol "X" yang menunjukkan bahwa suatu objek keluar dari sistem. Simbol ini ditempatkan di akhir lifeline untuk menandakan bahwa objek tersebut tidak lagi berinteraksi dengan sistem. Ini juga menunjukkan *destruction occurrence*, yaitu saat objek dihapus atau dihancurkan dari sistem.

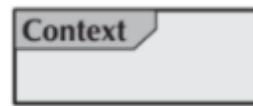
Contoh:



Keterangan:

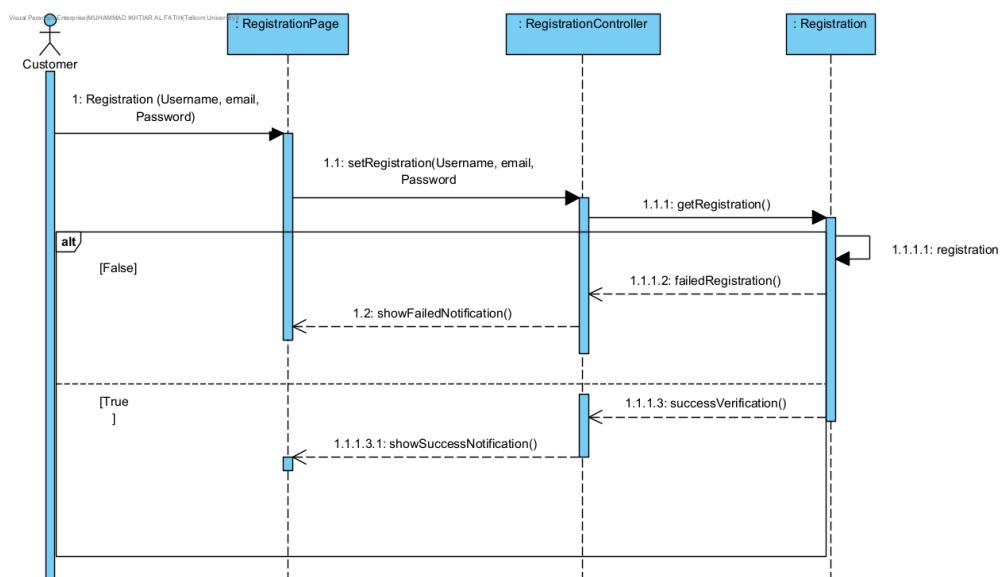
Dalam contoh yang diberikan, terdapat dua ilustrasi object destruction, di mana lifeline *:OrderController* dan *:Order* berhenti berkomunikasi dengan sistem setelah proses pemesanan selesai. Setelah pesanan ditampilkan di *:OrderPage*, kedua objek tersebut tidak lagi aktif dalam sistem.

6. Frame



Frame digunakan untuk menunjukkan konteks dalam diagram sequence. Frame menandai bagian sistem yang berhubungan dengan interaksi antar objek dalam diagram. Beberapa jenis frame dalam diagram sequence antara lain *Alt* (Alternatif) dan *Loop* (Perulangan).

Contoh:



Notasi Alt. Combined Fragment

Keterangan:

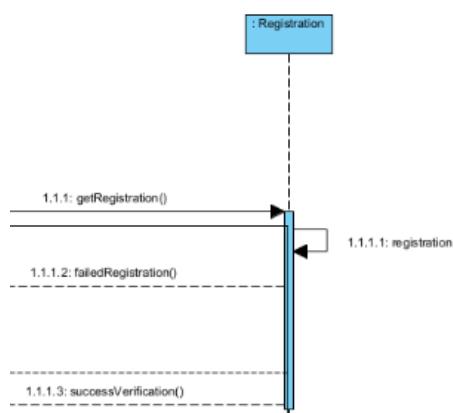
Pada contoh di atas, terdapat salah satu contoh dari frame yaitu notasi alt atau alternatif, di mana terdapat proses registrasi pengguna dalam sistem. Dalam proses ini, terdapat percabangan yang menentukan dua kemungkinan hasil, yaitu apakah registrasi berhasil atau gagal. Jika registrasi gagal, sistem akan mengirimkan pesan kegagalan dan menampilkan notifikasi kepada pengguna. Sebaliknya, jika registrasi berhasil, sistem akan mengirimkan pesan konfirmasi sukses dan menampilkan notifikasi keberhasilan. Kedua hasil ini akan menghasilkan kondisi dengan pesan yang dikirim sesuai dengan hasil verifikasi.

7. Self Message



Self Message adalah pesan yang dikirim oleh suatu objek kepada dirinya sendiri atau ke lifeline yang sama.

Contoh:



Keterangan:

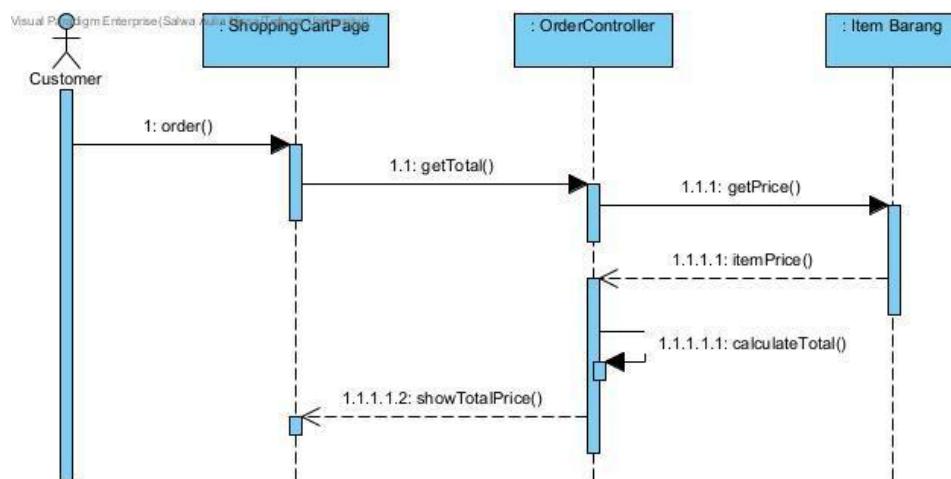
Pada gambar di atas, terdapat contoh dari self message pada lifeline : *Registration*, di mana terdapat pesan *registration()* yang menunjukkan bahwa sistem melakukan proses pendaftaran secara internal. Setelah menerima permintaan registrasi melalui pesan *getRegistration()*, sistem akan memvalidasi data yang diberikan sebelum mengembalikan hasil. Jika validasi gagal, pesan *failedRegistration()* akan dikirimkan, sedangkan jika validasi berhasil, sistem akan mengirimkan pesan *successVerification()* untuk melanjutkan proses registrasi ke tahap berikutnya.

8. Recursive Message



Recursive Message adalah pesan yang memicu pemanggilan fungsi atau metode berulang pada objek yang sama.

Contoh:



Notasi Recursive Message

Keterangan:

Pada gambar di atas, terdapat contoh recursive message dengan pesan `calculateTotal()`. Dalam diagram ini, setelah mendapatkan harga barang dari entitas `ItemBarang`, lifeline `OrderController` mengirimkan pesan `calculateTotal()` ke dirinya sendiri. Pesan ini digunakan untuk menghitung total harga akhir dari semua item yang ada di dalam shopping cart, sehingga sistem dapat menentukan jumlah yang harus dibayarkan oleh pengguna.

3. Entity-Controller-Boundary (ECB)

Entity-Controller-Boundary (ECB) adalah pola arsitektur perangkat lunak yang digunakan dalam pengembangan aplikasi berorientasi objek. Pola ini membantu memperjelas struktur dan alur kerja sistem dalam diagram sequence dengan membagi sistem ke dalam tiga bagian utama:

1. Entity: Menyimpan data dan menangani logika bisnis.
2. Controller: Mengatur proses, mengendalikan alur kerja, dan menghubungkan entity dengan boundary.
3. Boundary: Berperan sebagai antarmuka antara sistem dan pengguna atau sistem eksternal, menerima input serta menampilkan output.

Sebagai contoh, dalam sebuah diagram sequence, setelah mendapatkan harga barang (*item price*) dari *entity ItemBarang*, *lifeline OrderController* akan mengirim pesan `calculateTotal()` ke dirinya sendiri untuk menghitung total harga semua item dalam keranjang belanja (*shopping cart*).

4. Pedoman Membuat Sequence Diagram

1. Identifikasi Aktor dan Objek

Tentukan semua pihak yang Langkah pertama dalam menyusun diagram sequence adalah mengidentifikasi semua aktor dan objek yang terlibat dalam sistem. Aktor bisa berupa pengguna manusia atau sistem eksternal yang berinteraksi dengan sistem utama, sedangkan objek adalah komponen dalam sistem yang menerima dan mengirim pesan. Dalam diagram sequence, aktor dan objek disusun dari kiri ke kanan berdasarkan urutan keterlibatan mereka dalam skenario use case. Aktor atau objek yang pertama kali memulai interaksi ditempatkan di posisi paling kiri untuk mencerminkan alur interaksi yang logis dan mudah dipahami.

2. Gunakan Notasi yang Tepat

Setelah aktor dan objek diidentifikasi, langkah berikutnya adalah menggunakan notasi yang sesuai agar diagram lebih mudah dibaca dan dipahami. Aktor biasanya digambarkan sebagai ikon manusia, sedangkan objek direpresentasikan dengan persegi panjang yang berisi nama objek tersebut. Hubungan antar aktor dan objek digambarkan dengan garis dan panah untuk menunjukkan alur komunikasi atau pesan yang dikirim. Menggunakan notasi yang standar membantu menjaga konsistensi dan memudahkan interpretasi diagram oleh berbagai pemangku kepentingan.

3. Membuat Pesan yang Jelas dan Ringkas

Setiap pesan yang dikirim dalam diagram harus memiliki nama yang jelas dan deskriptif agar dapat menjelaskan tindakan atau informasi yang dikirim. Misalnya, daripada hanya menggunakan "*Login()*", lebih baik menggunakan "*LoginUser()*" untuk memperjelas maksud dari pesan tersebut. Jika sebuah pesan memiliki nilai yang dikembalikan dan tidak langsung jelas, sebaiknya tambahkan return message agar lebih eksplisit, seperti "*ValidasiBerhasil()*" untuk menunjukkan bahwa validasi telah berhasil dilakukan. Nama pesan dan return message juga perlu ditempatkan di dekat ujung panah masing-masing agar lebih mudah dipahami.

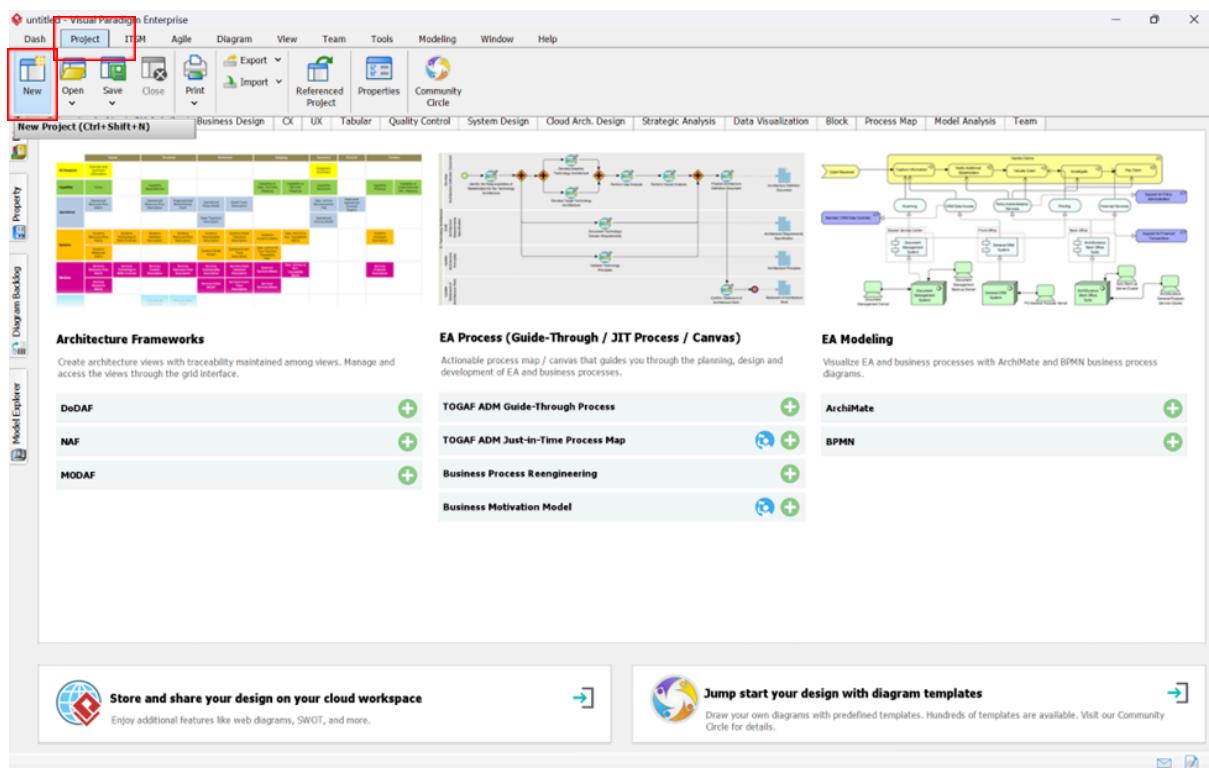
4. Menyusun Pesan Berdasarkan Urutan

Agar diagram lebih terstruktur dan mudah dibaca, pesan-pesan harus disusun secara berurutan dari kiri ke kanan, mengikuti kebiasaan membaca. Penyusunan ini membantu menghindari kebingungan dalam memahami alur komunikasi antar objek. Selain itu, pesan harus tetap berada di bawah objek yang mengirim atau menerimanya untuk menjaga keteraturan diagram. Dengan mengikuti aturan ini, diagram sequence akan lebih terorganisir dan mudah dipahami oleh pengembang serta pihak terkait lainnya.

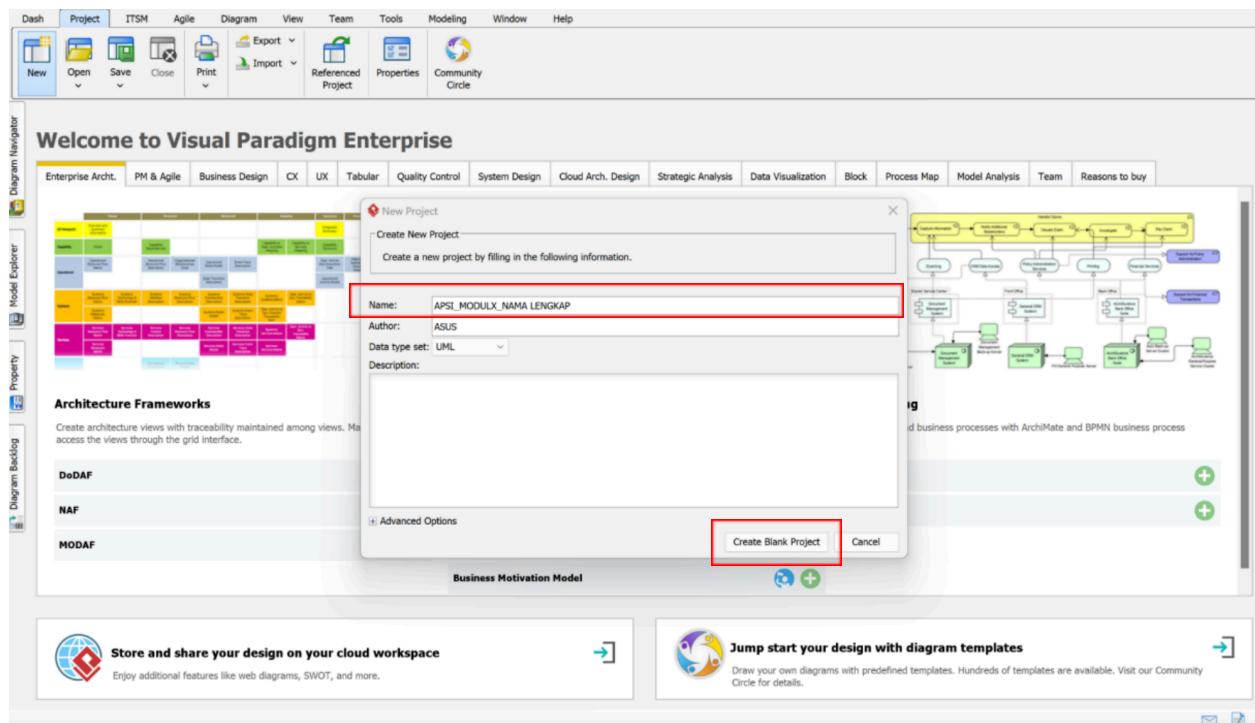
LANGKAH-LANGKAH PRAKTIKUM

A. Cara Membuat Class Diagram

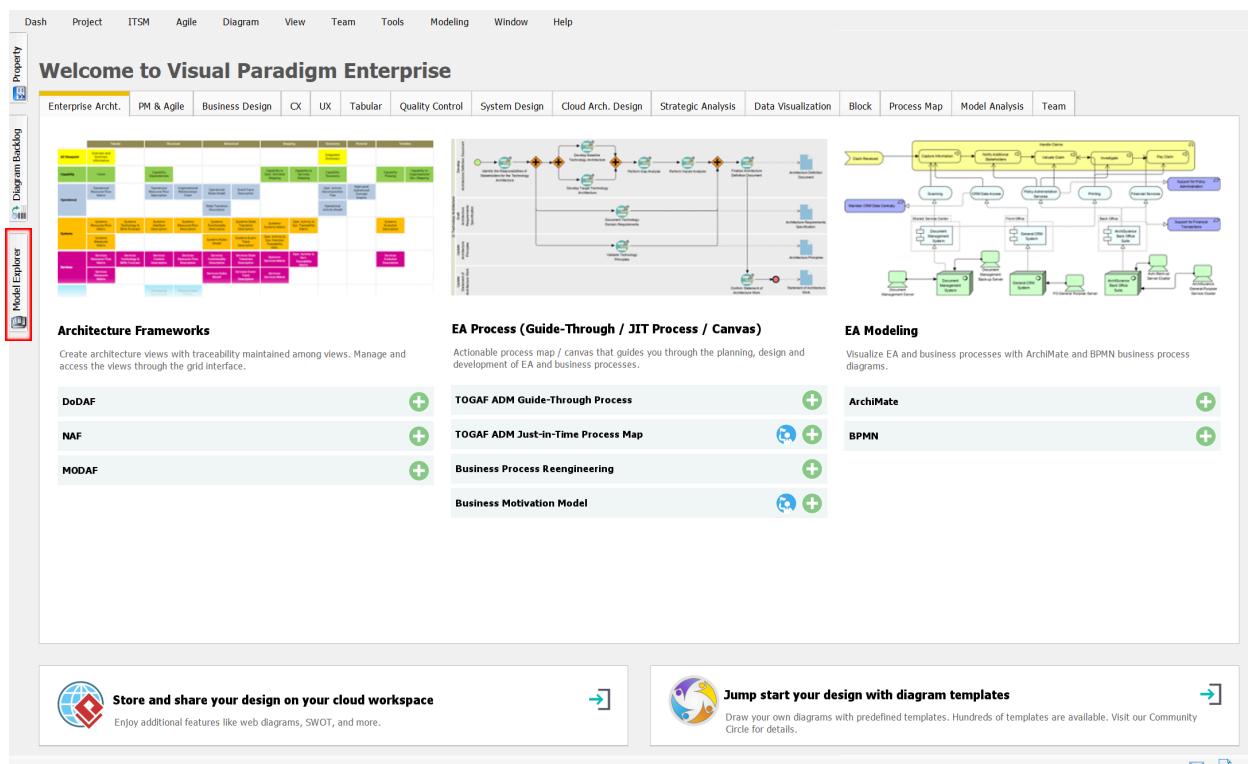
1. Membuka Visual Paradigm
2. Pada tampilan awal Visual Paradigm, klik Project > klik New



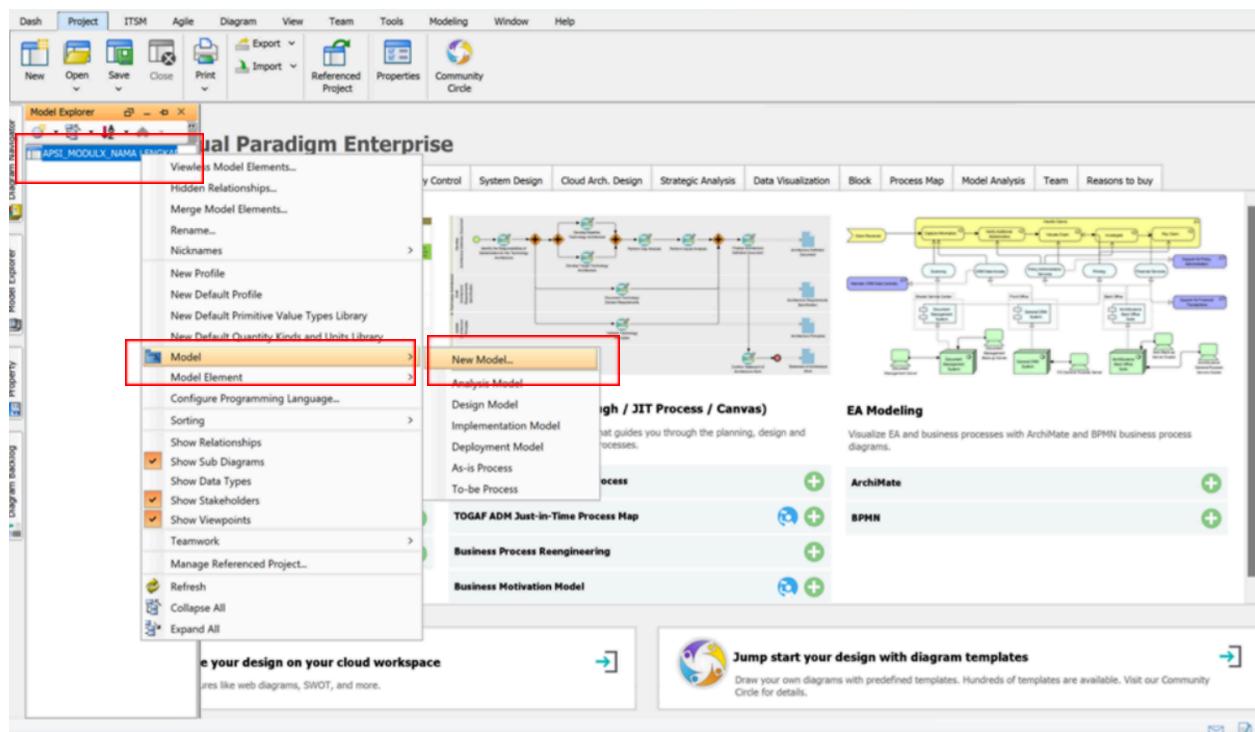
3. Isi Name sesuai dengan ketentuan , kemudian klik Create Blank Project.



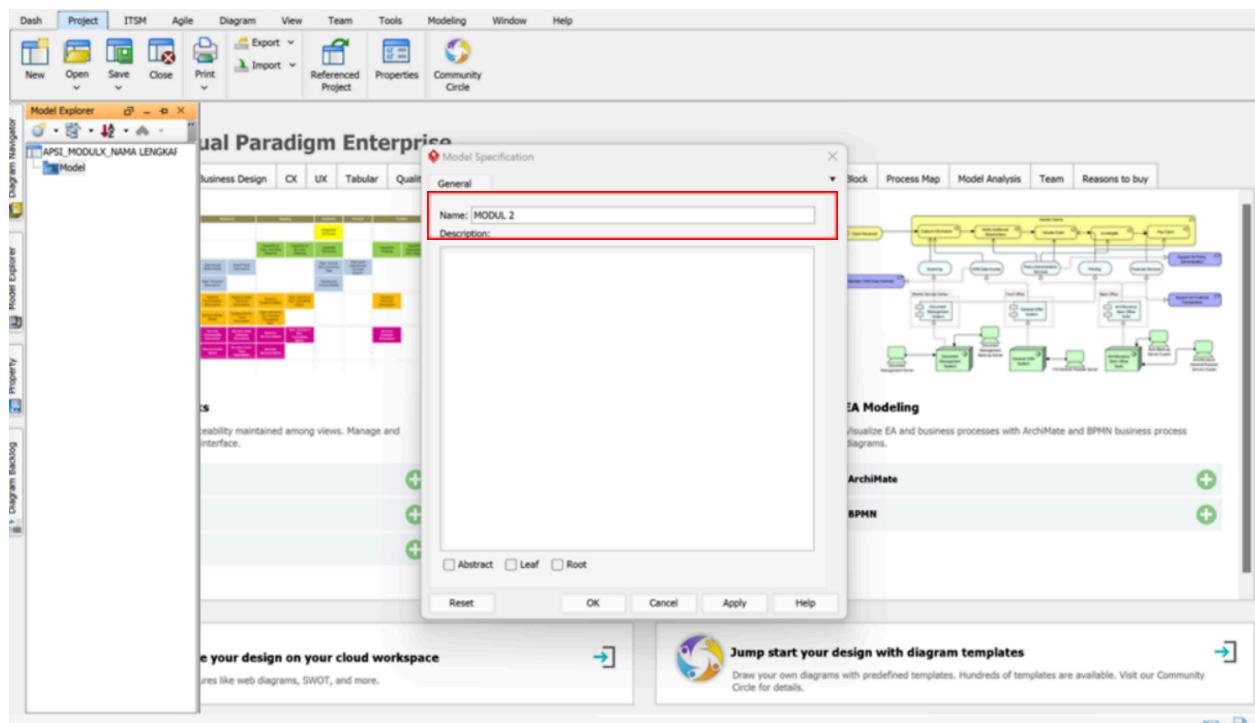
4. Buka Model Explorer pada sisi kiri halaman.



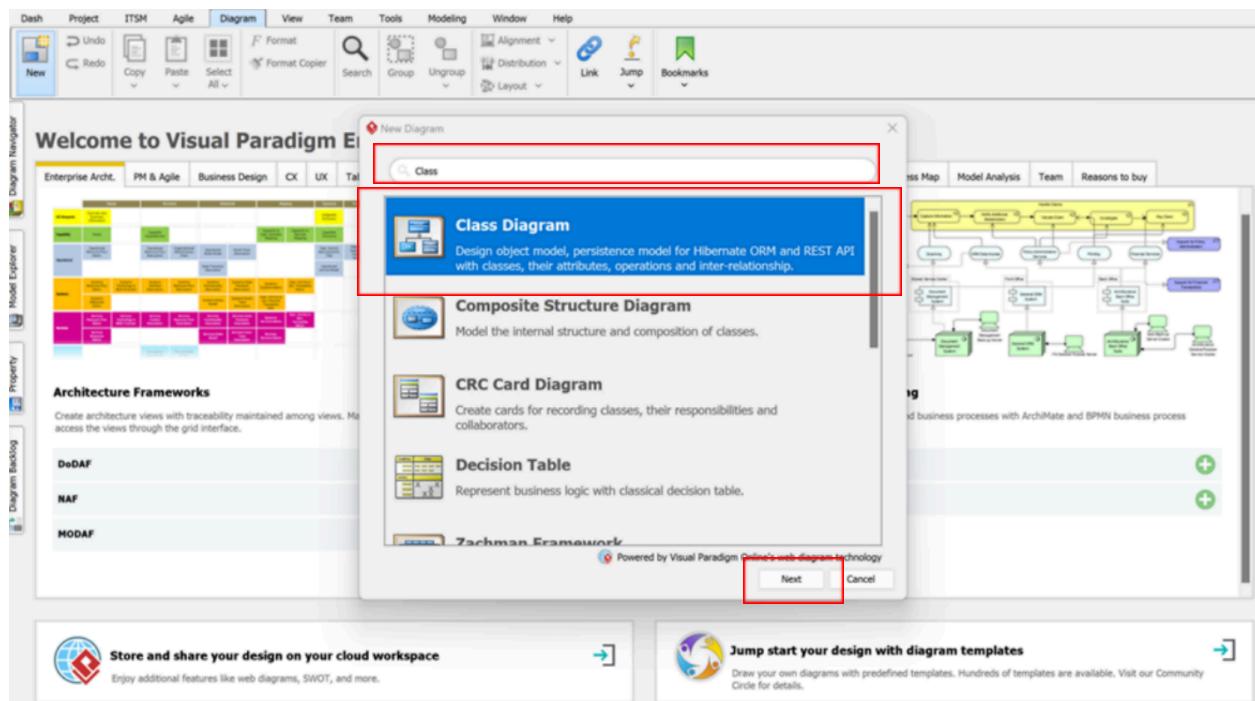
5. Klik kanan pada Project yang telah dibuat > klik Model > klik New Model.



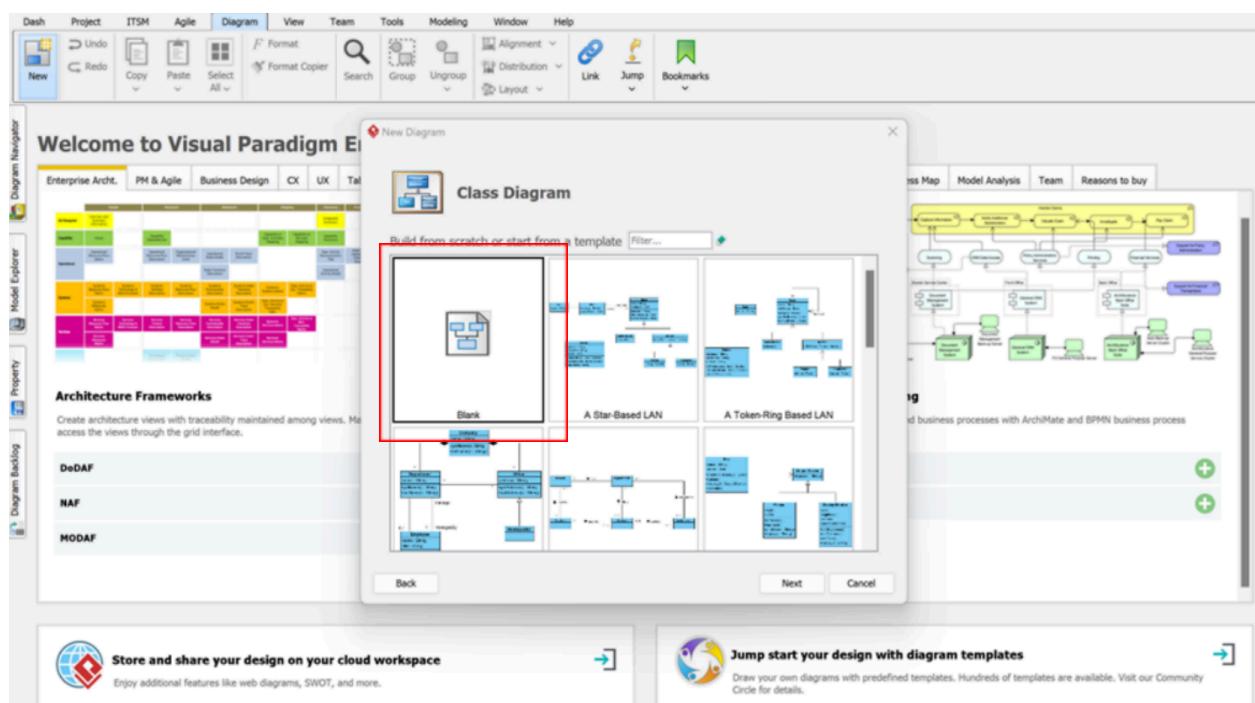
6. Isi Name sesuai modul yang akan dikerjakan.



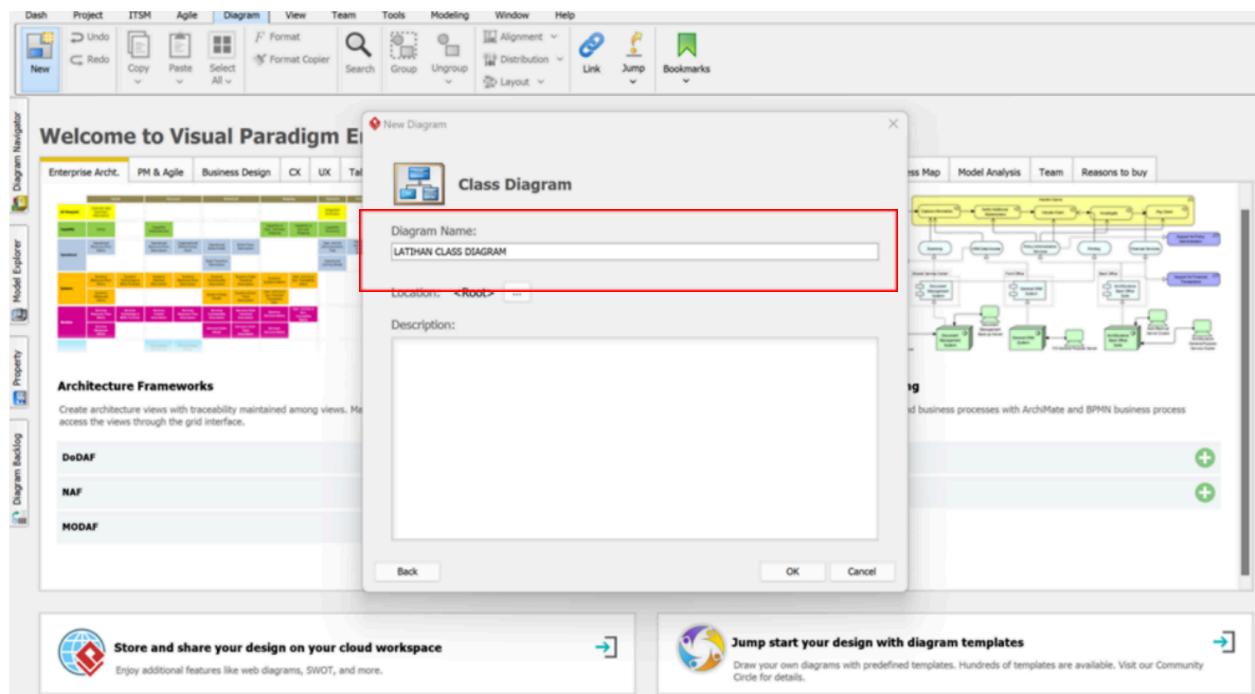
7. Pilih Tab Diagram > pilih New > Cari Class Diagram > Klik Next.



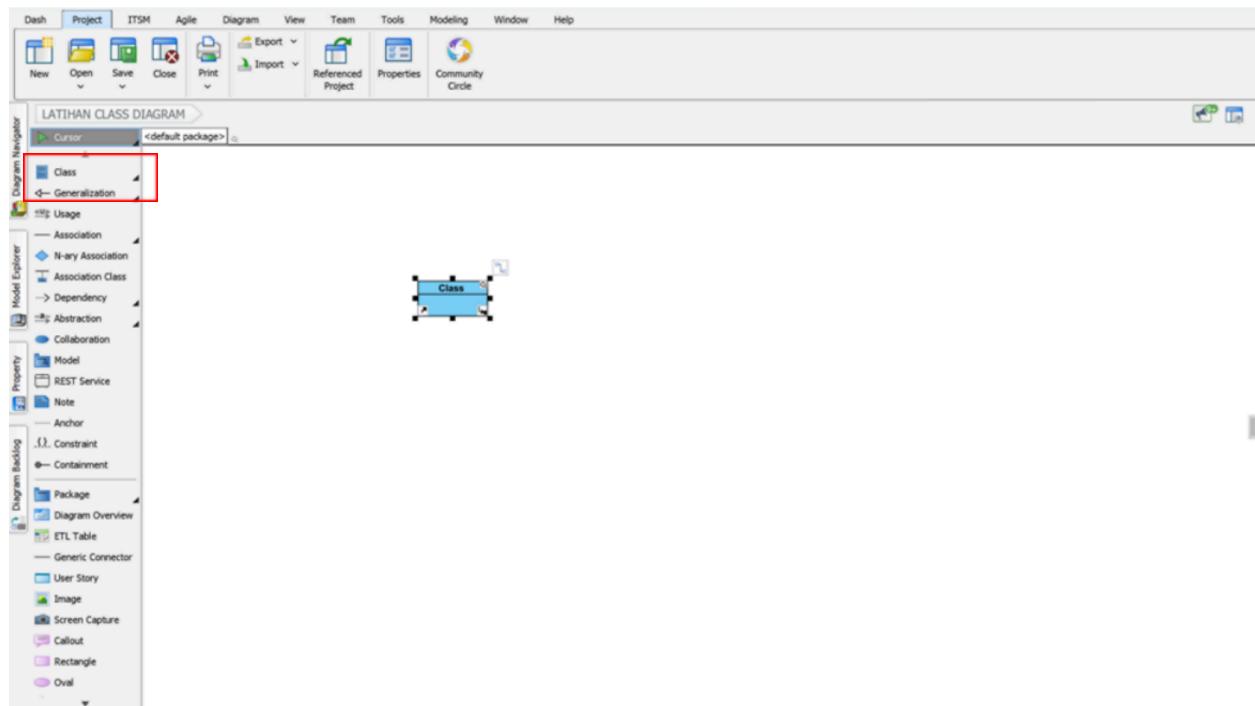
8. Pilih Blank.



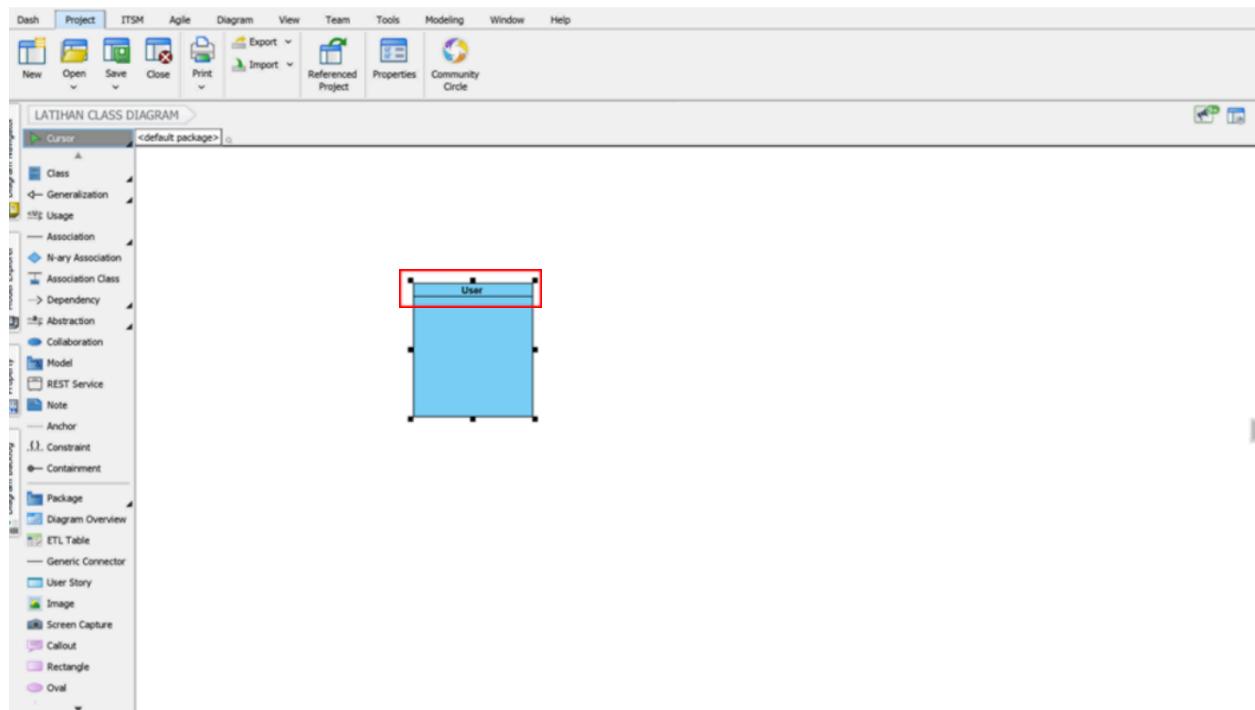
9. Isi Name sesuai ketentuan.



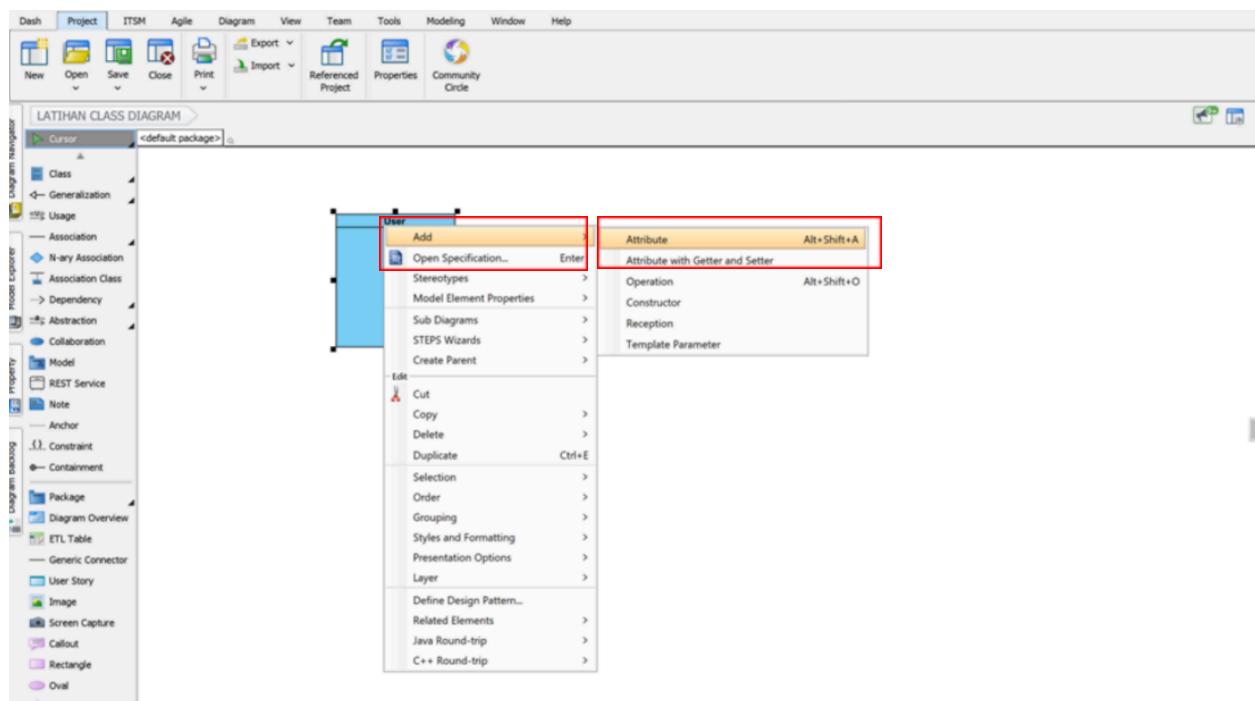
10. Klik Class.



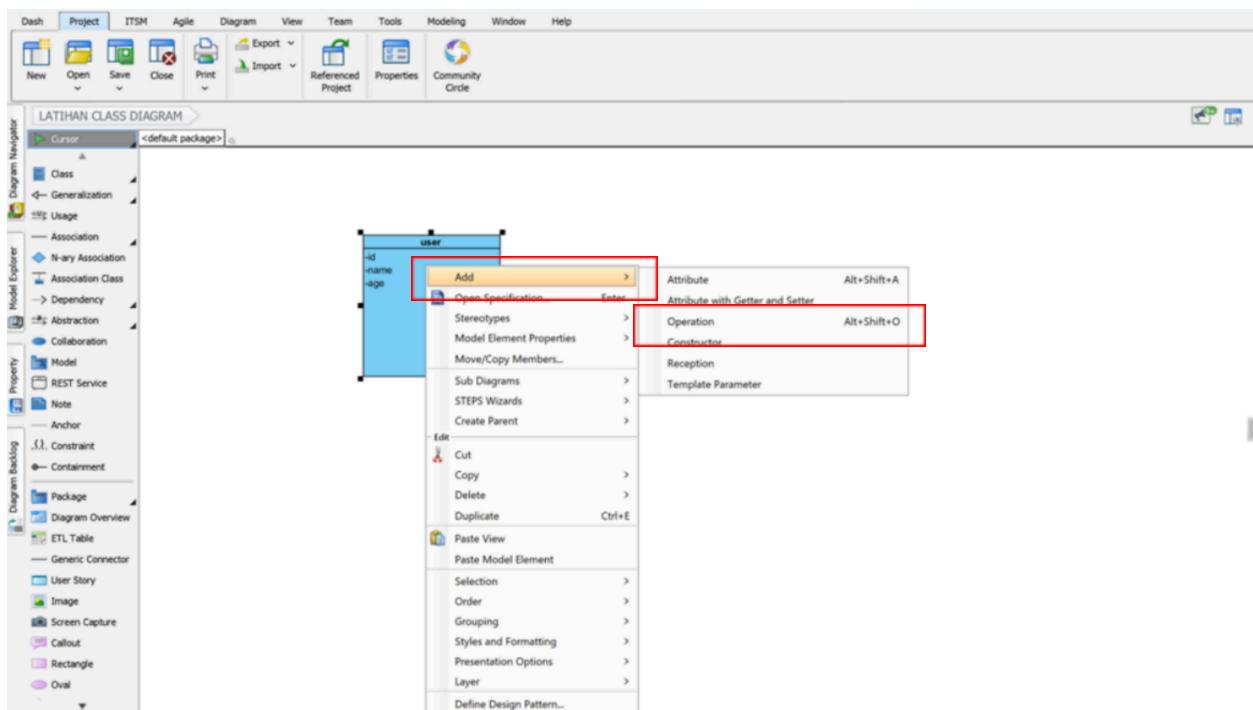
11. Klik dua kali pada nama Class dan ubah nama Class sesuai dengan ketentuan.



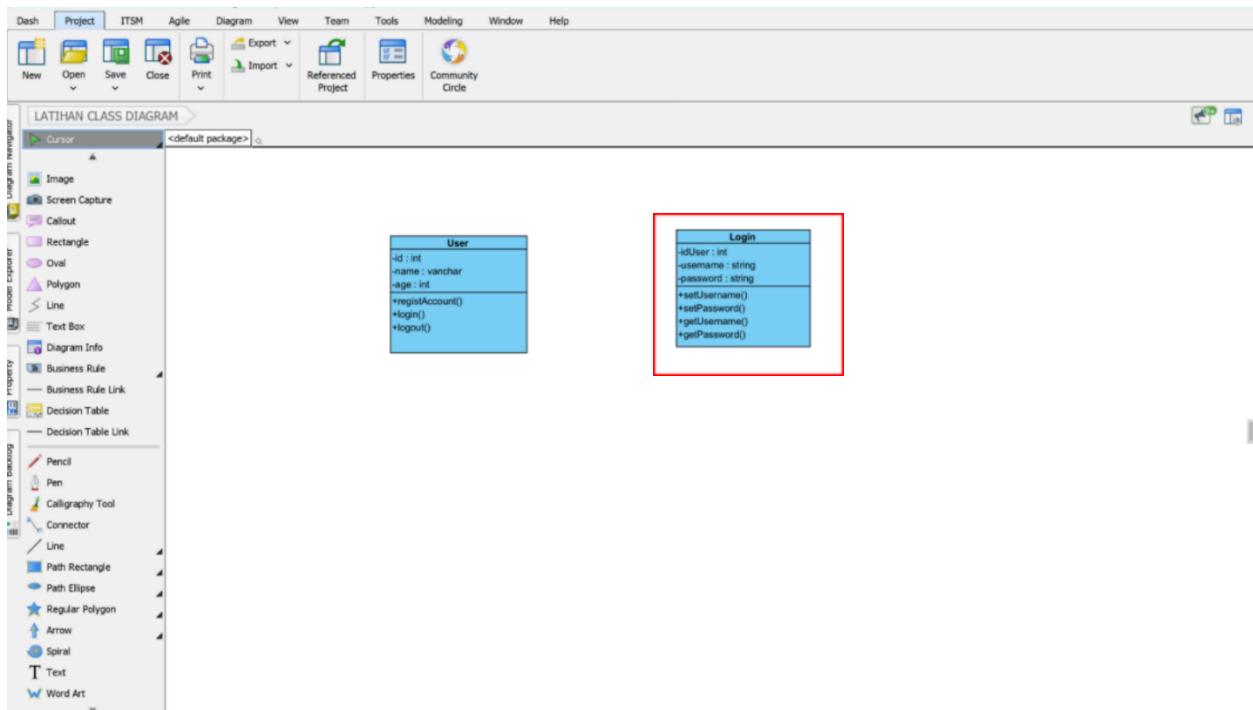
12. Klik kanan pada Class > Klik Add > Klik Atribute untuk menambahkan atribut.



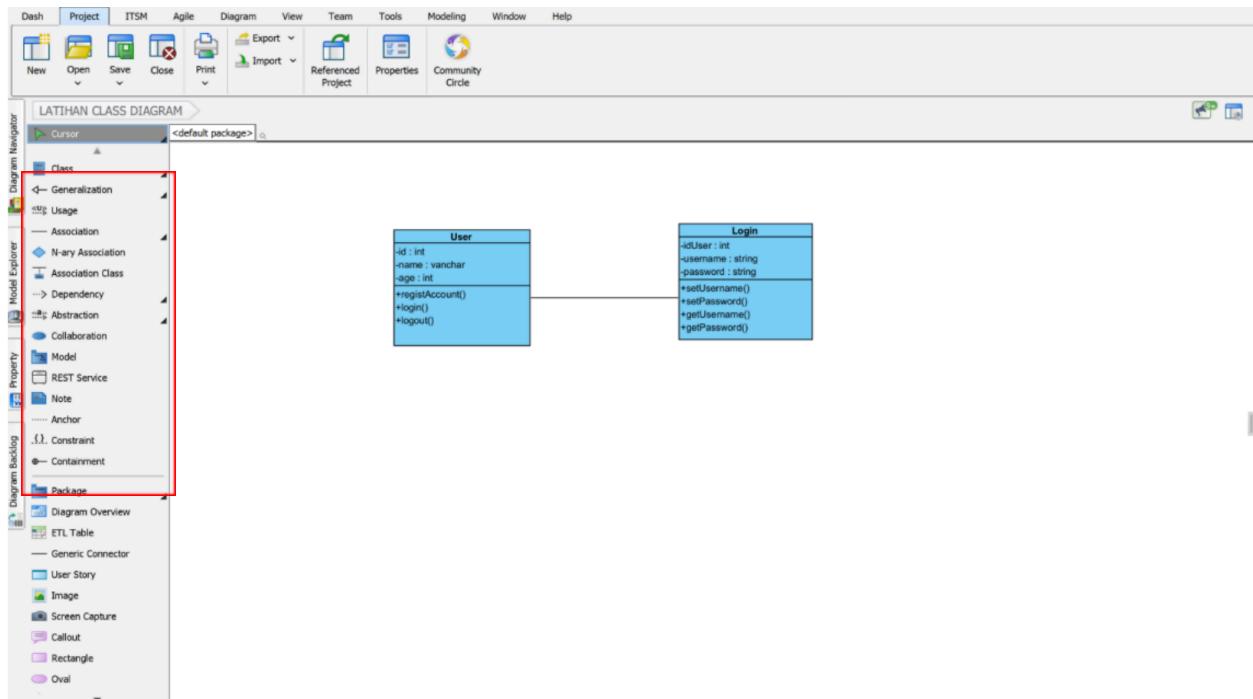
13. Klik kanan pada Class > Klik Add > Klik Operation untuk menambahkan operasi.



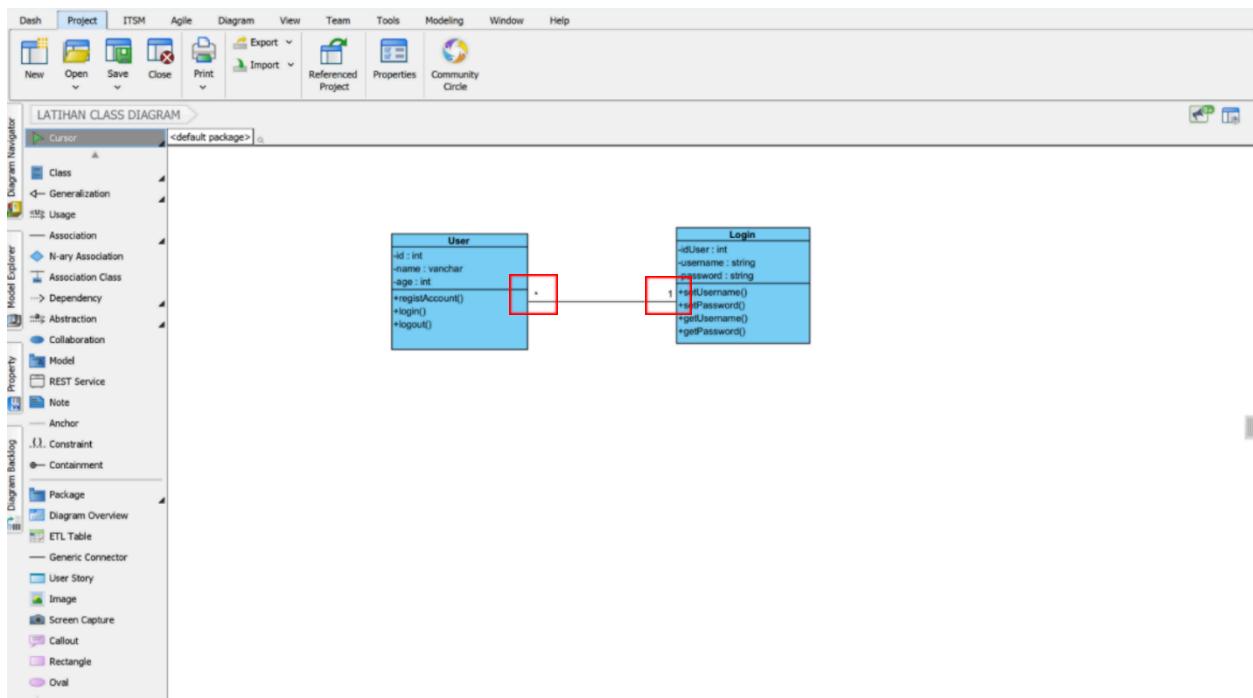
14. Buat Class diagram yang lain sebelum lanjut ke poin berikutnya.



15. Pilih relasi yang sesuai untuk menghubungkan antar Class.

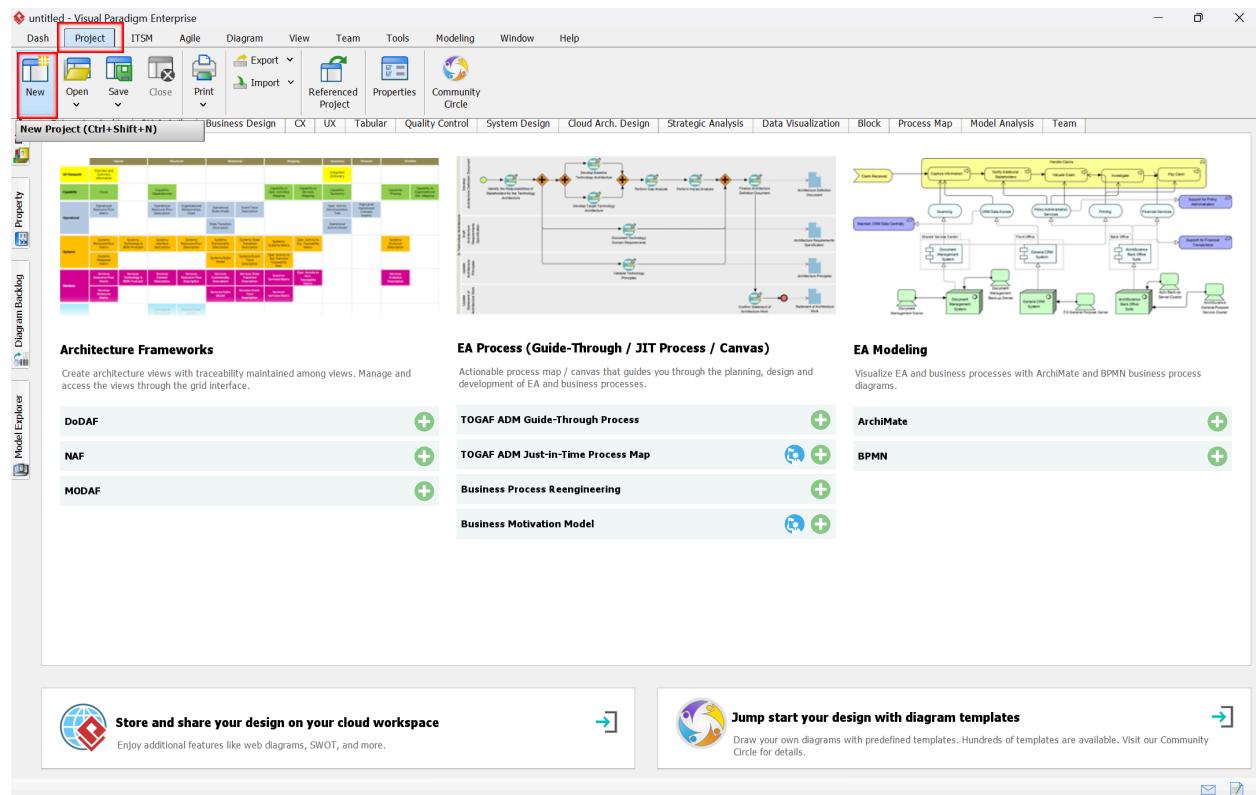


16. Double klik pada setiap ujung relasi untuk memasukan Kardinalitas pada setiap Class yang terhubung.

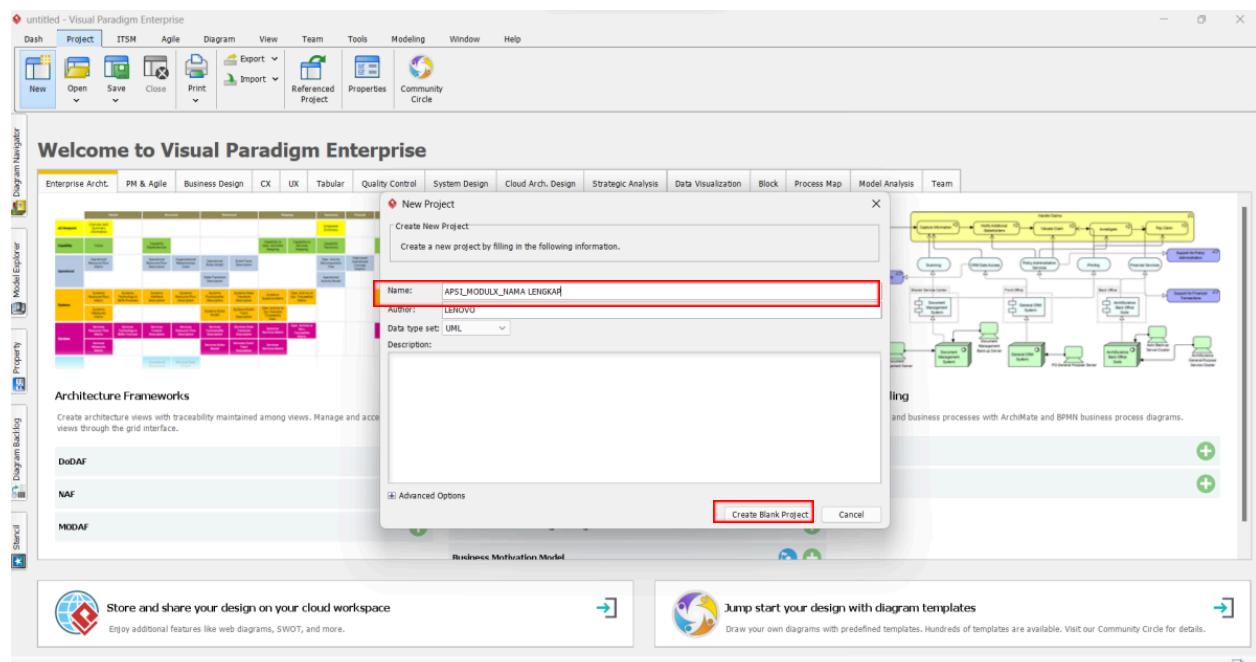


B. Cara Membuat Sequence Diagram

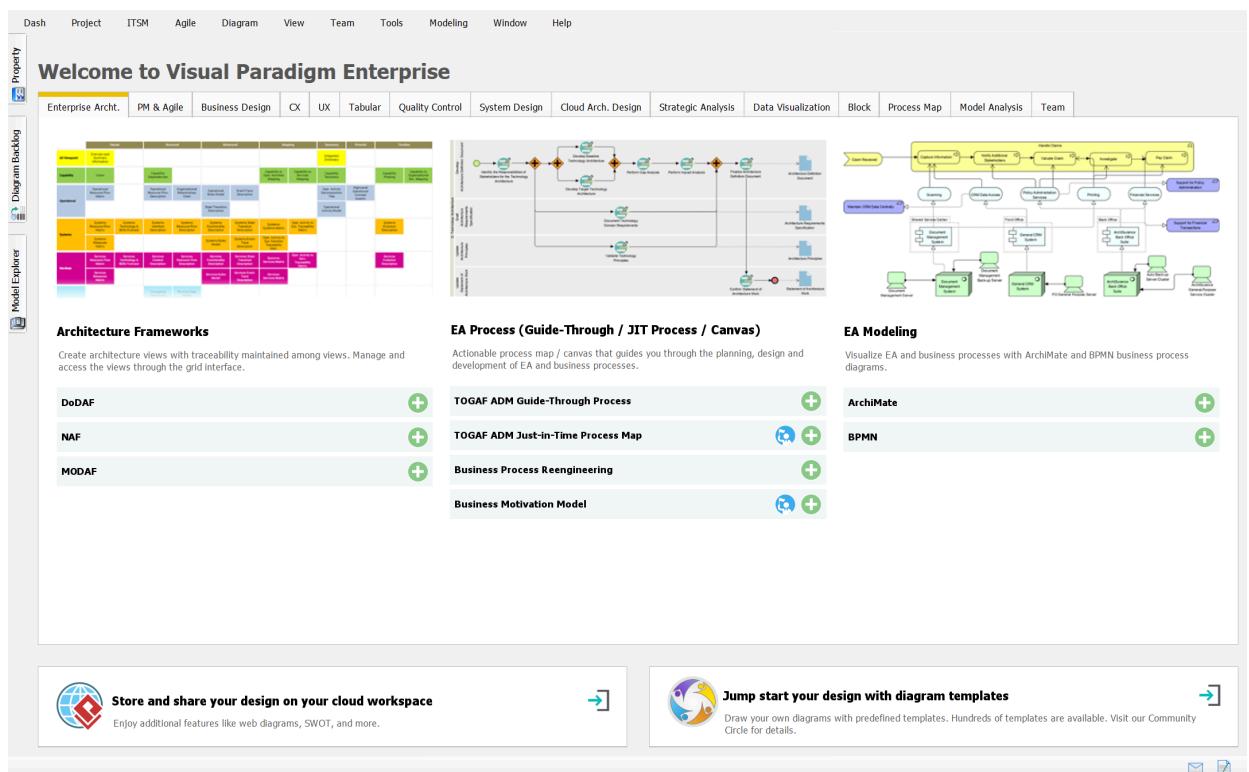
- Pada tampilan awal *Dashboard Visual Paradigm*, klik *Project* lalu klik *New*.



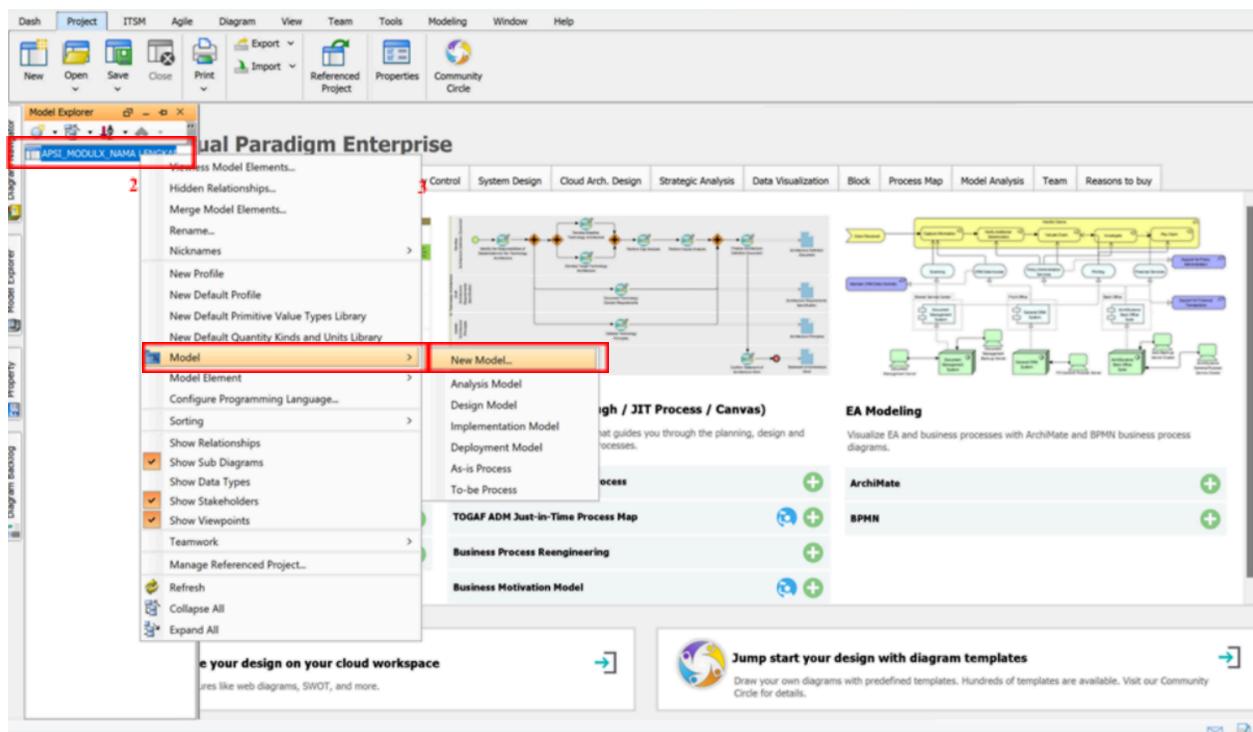
- Isi nama sesuai ketentuan kemudian klik *Create Blank Project*.



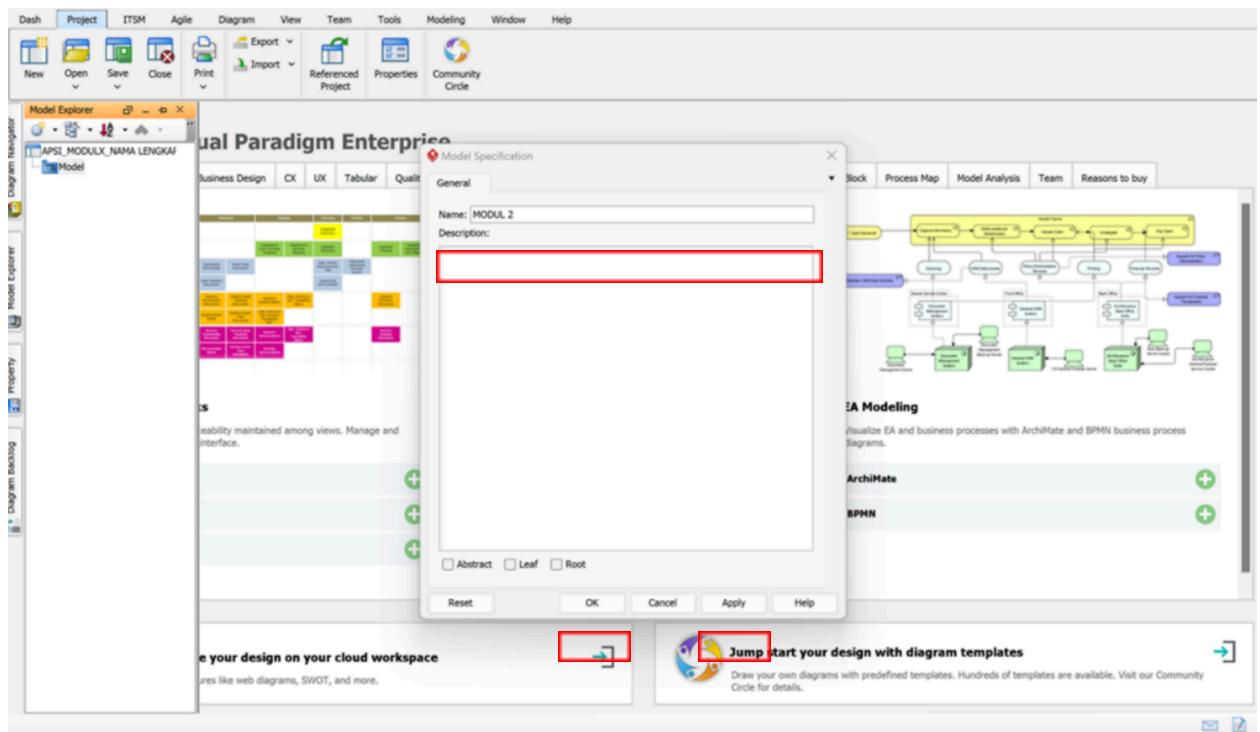
3. Buka Model Explorer pada sisi kiri halaman.



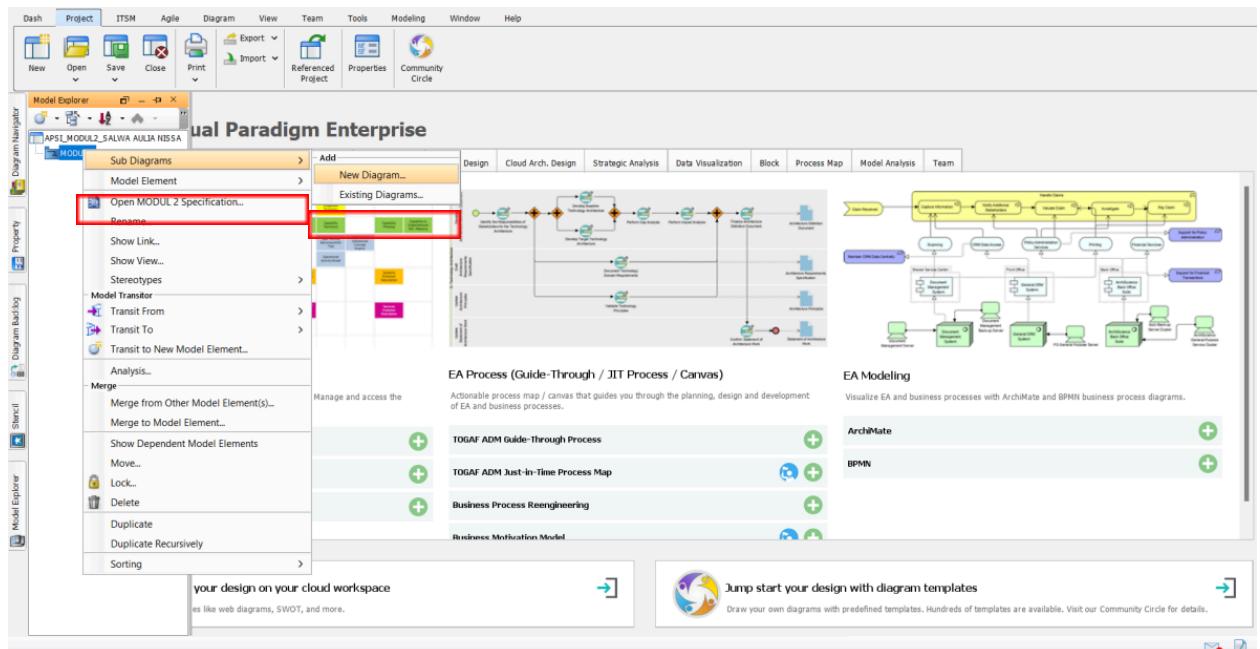
4. Klik kanan pada Project yang telah dibuat > klik Model > klik New Model.



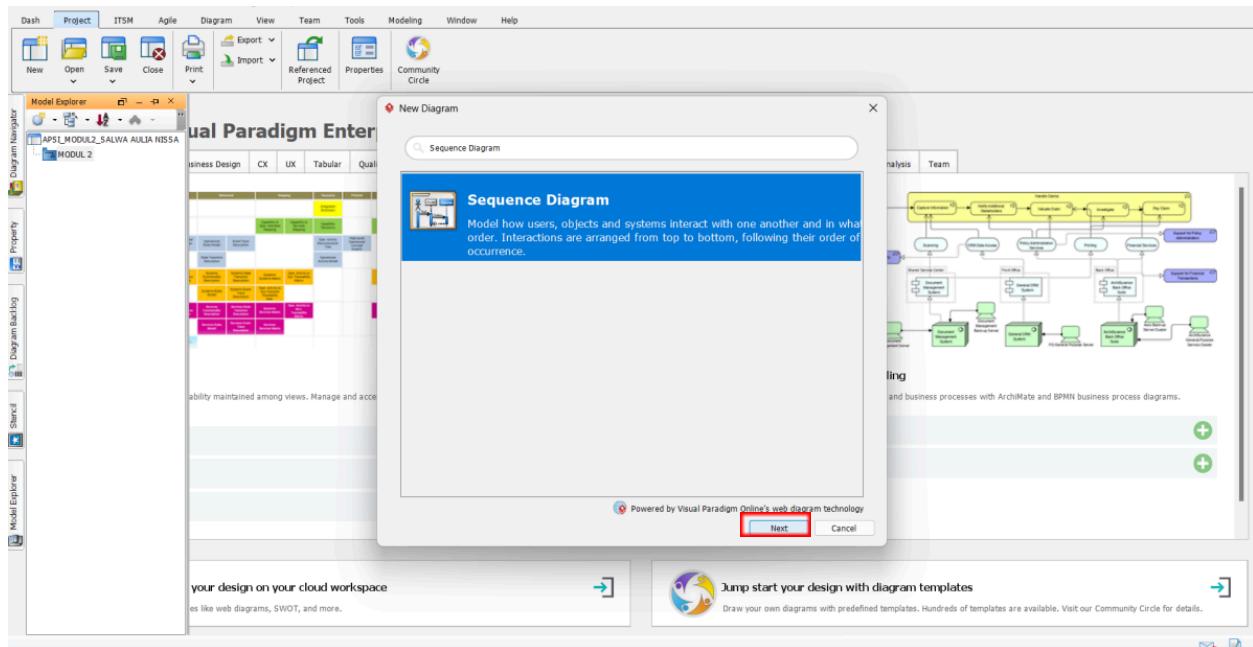
5. Isi Name sesuai dengan modul yang dikerjakan, lalu klik Apply, lalu klik OK.



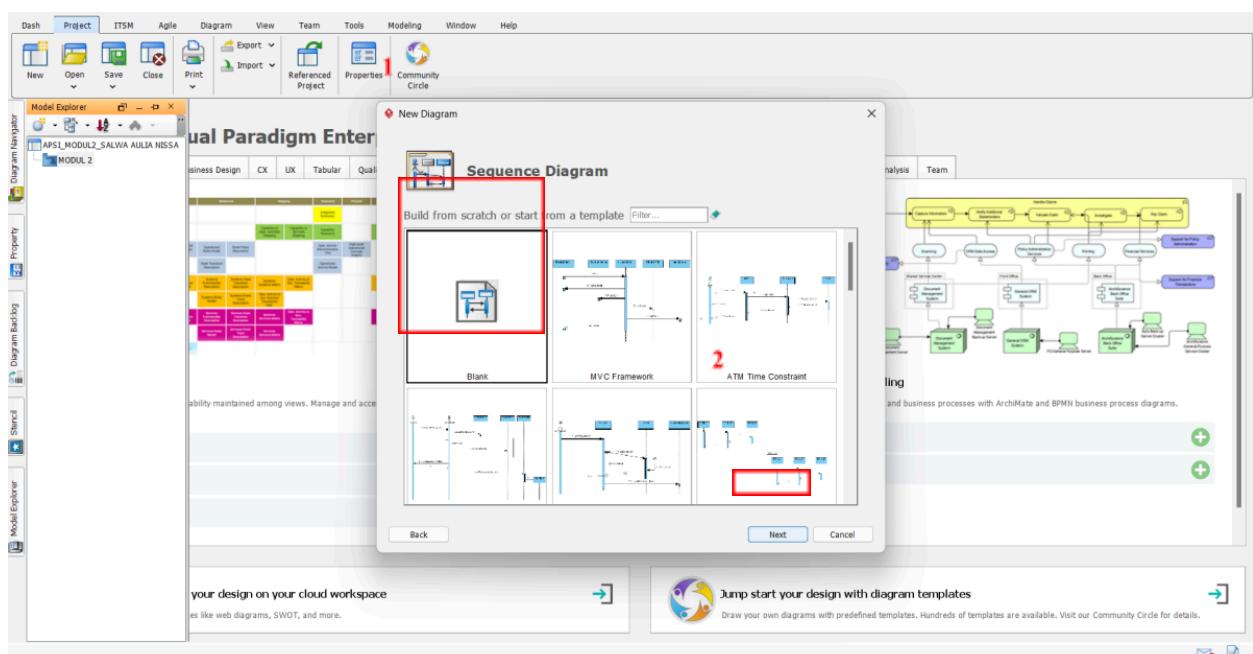
6. Klik kanan pada model yang telah dibuat > klik Sub Diagram > New Diagram.



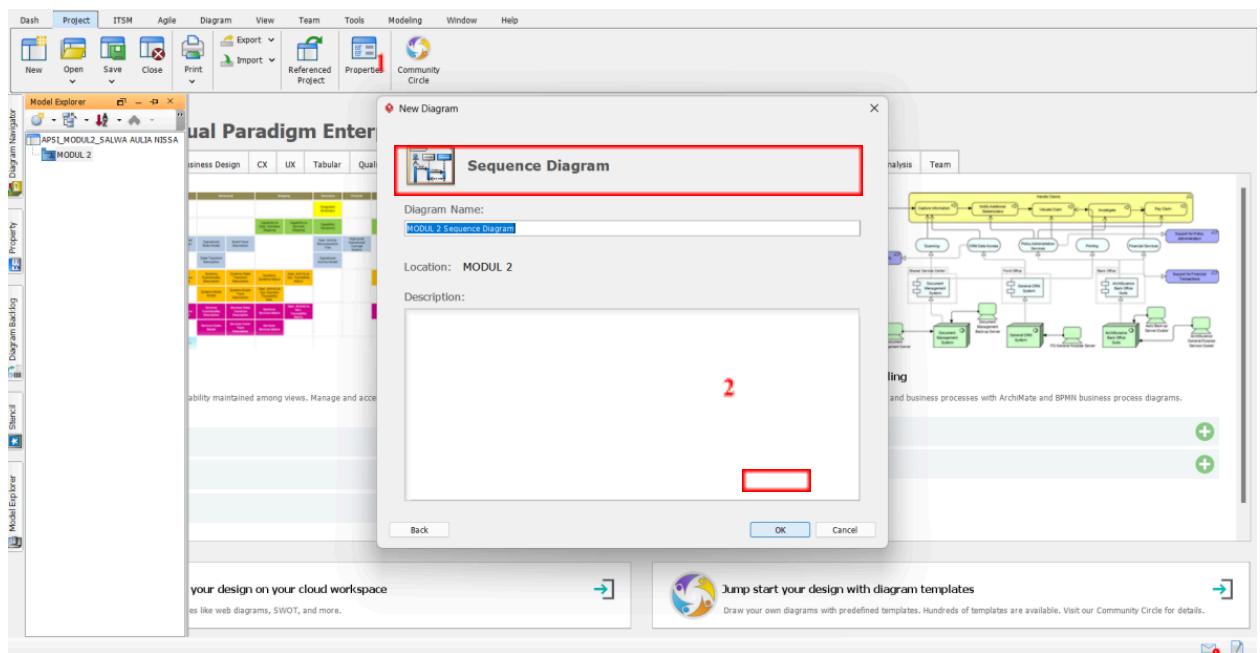
7. Cari “Sequence Diagram” di pencarian, lalu klik Next.



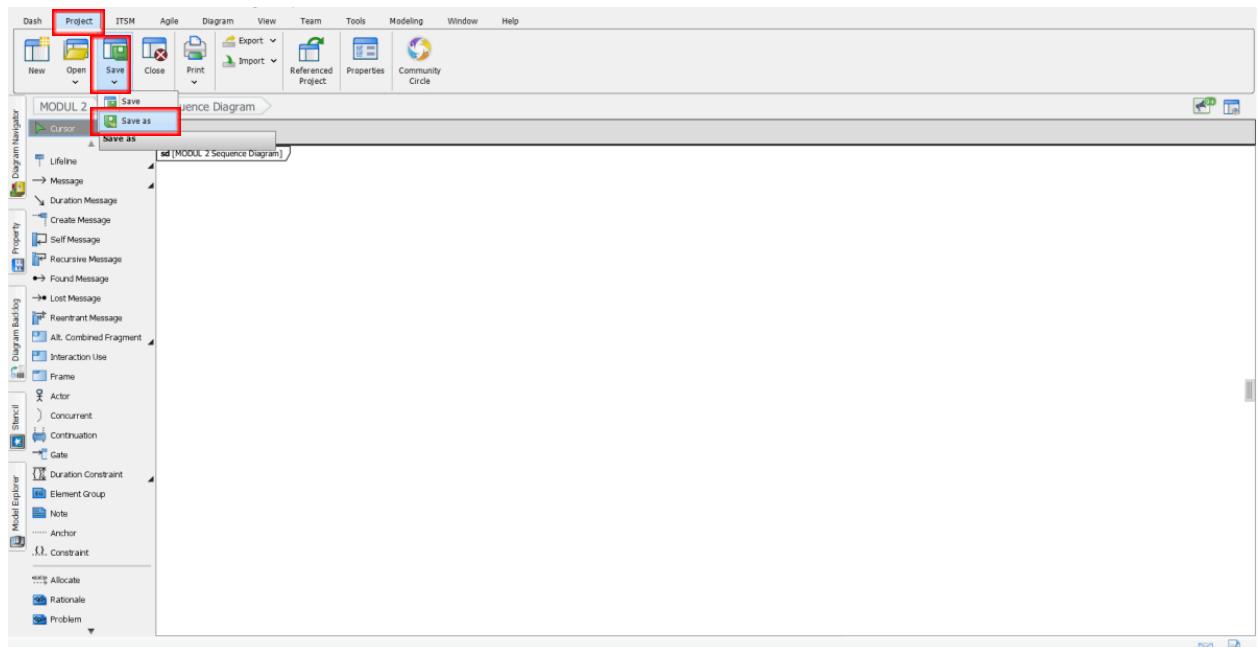
8. Klik “Blank” lalu klik Next



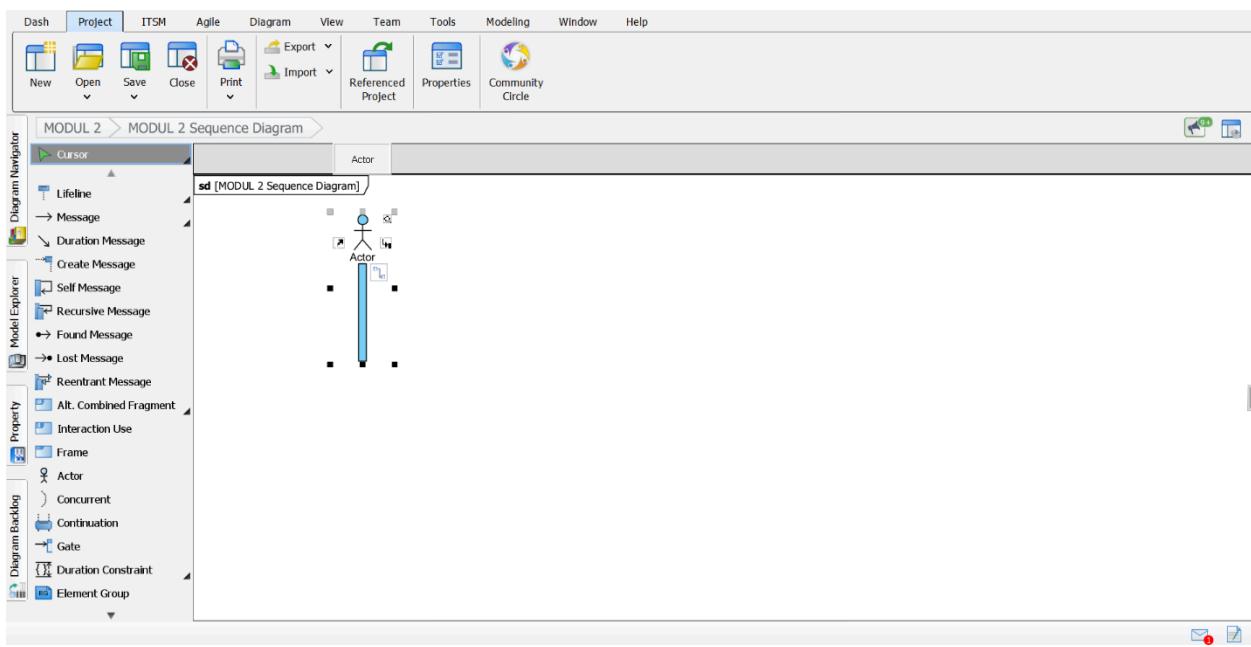
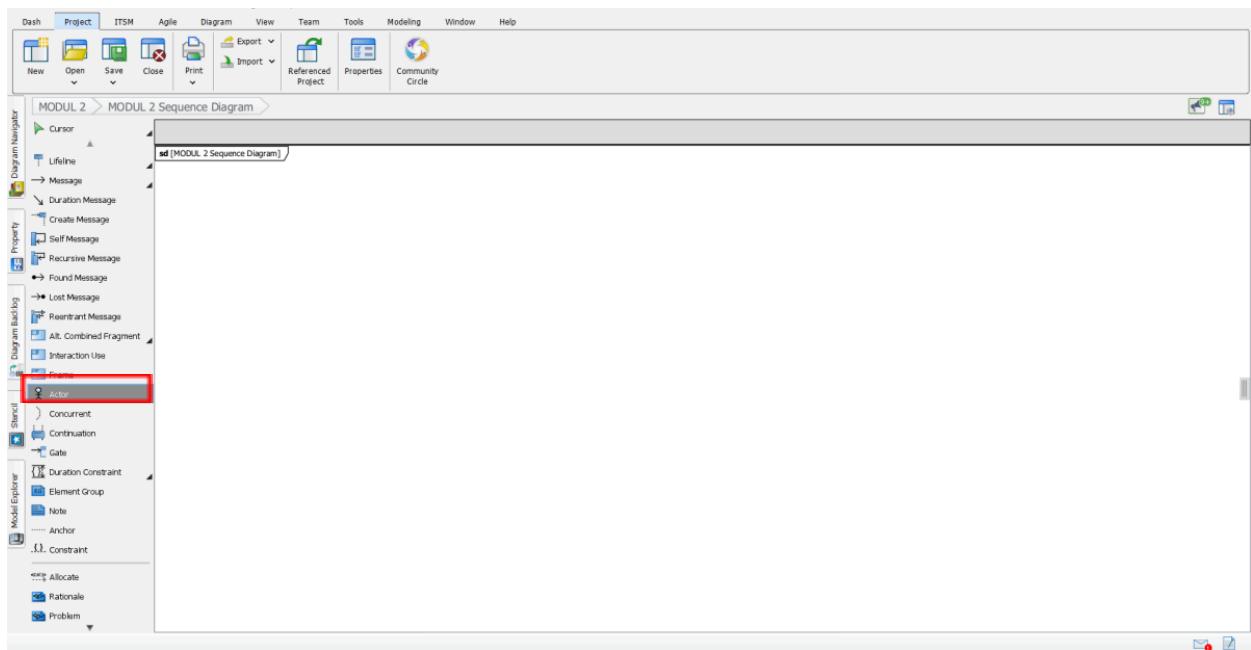
9. Berikan nama sesuai modul dan diagram yang dikerjakan. Lalu klik *Ok*.



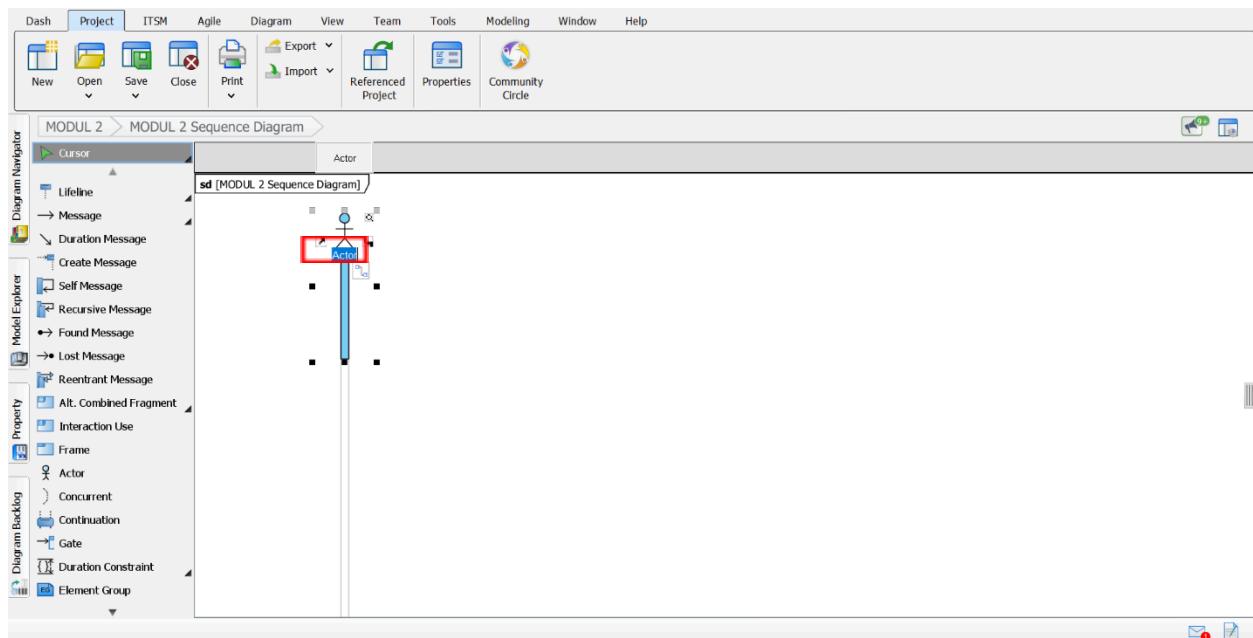
10. Sebelum memulai penggeraan, pastikan menyimpan pada laptop masing - masing, dengan klik *Project > Save as*.



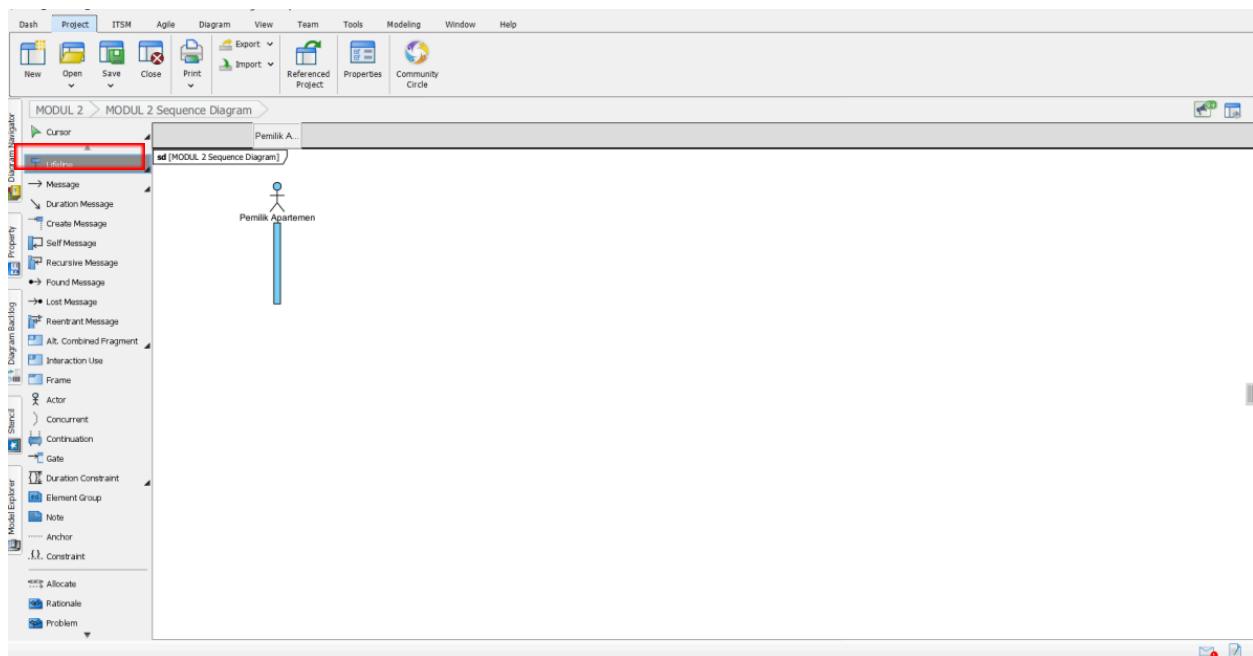
11. Dari toolbar pilih Actor dan lakukan *drag and drop* untuk ditambahkan ke worksheet.

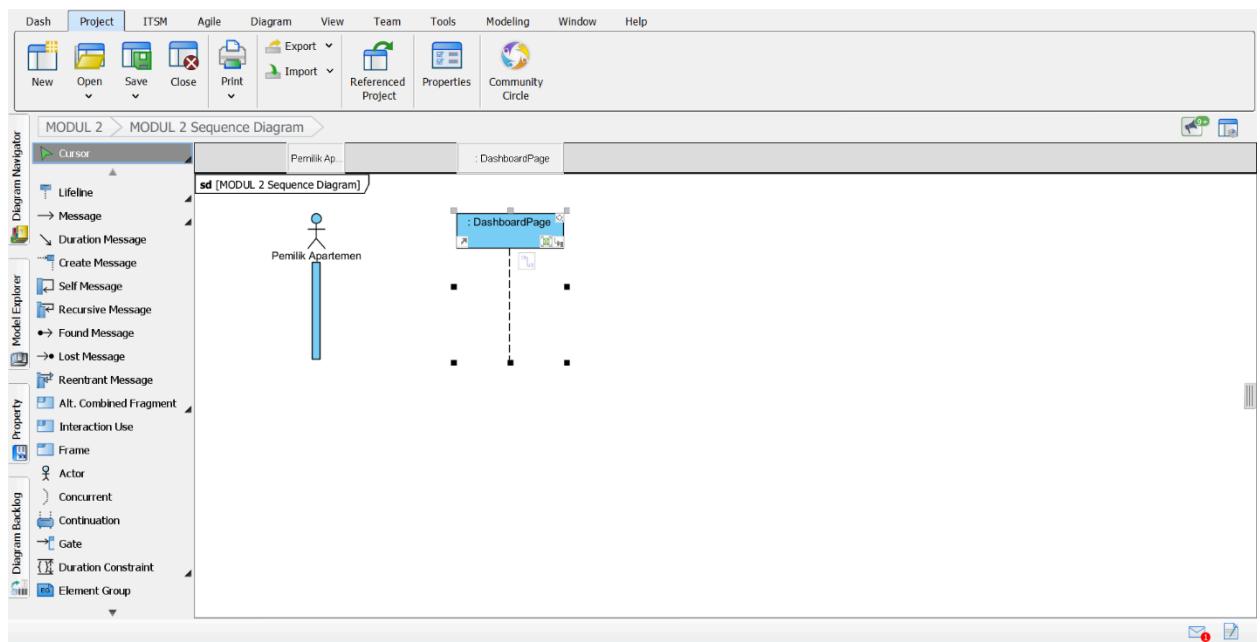


12. Untuk mengganti nama Actor, double-click Actor dan ganti namanya sesuai skenario studi kasus.

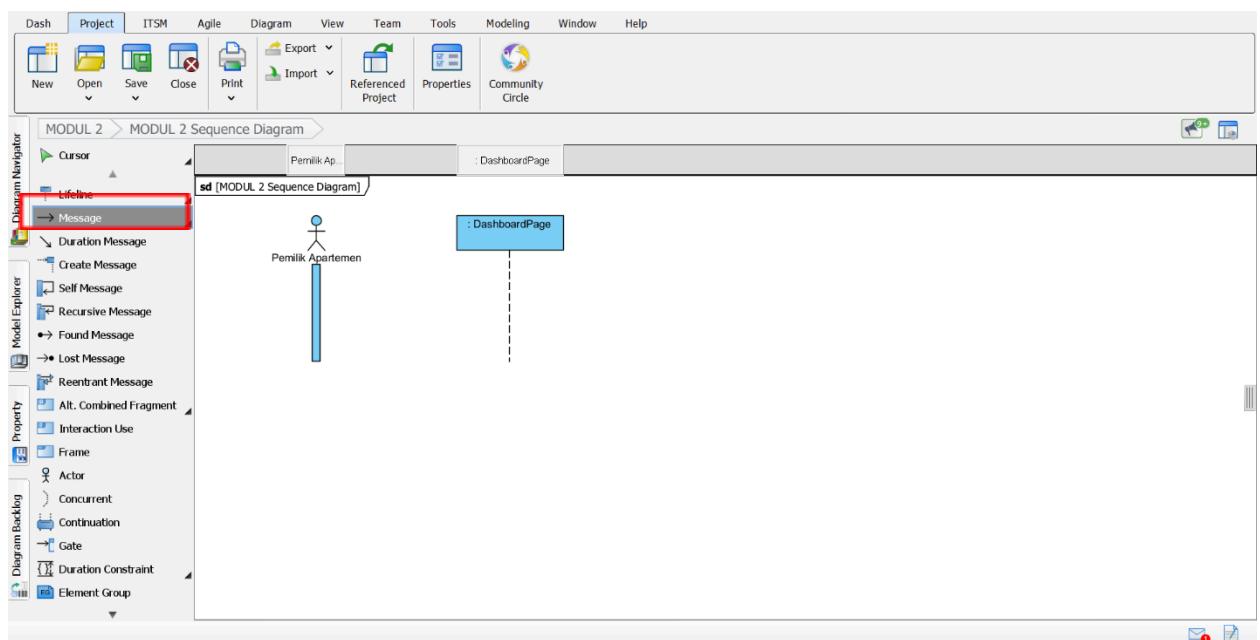


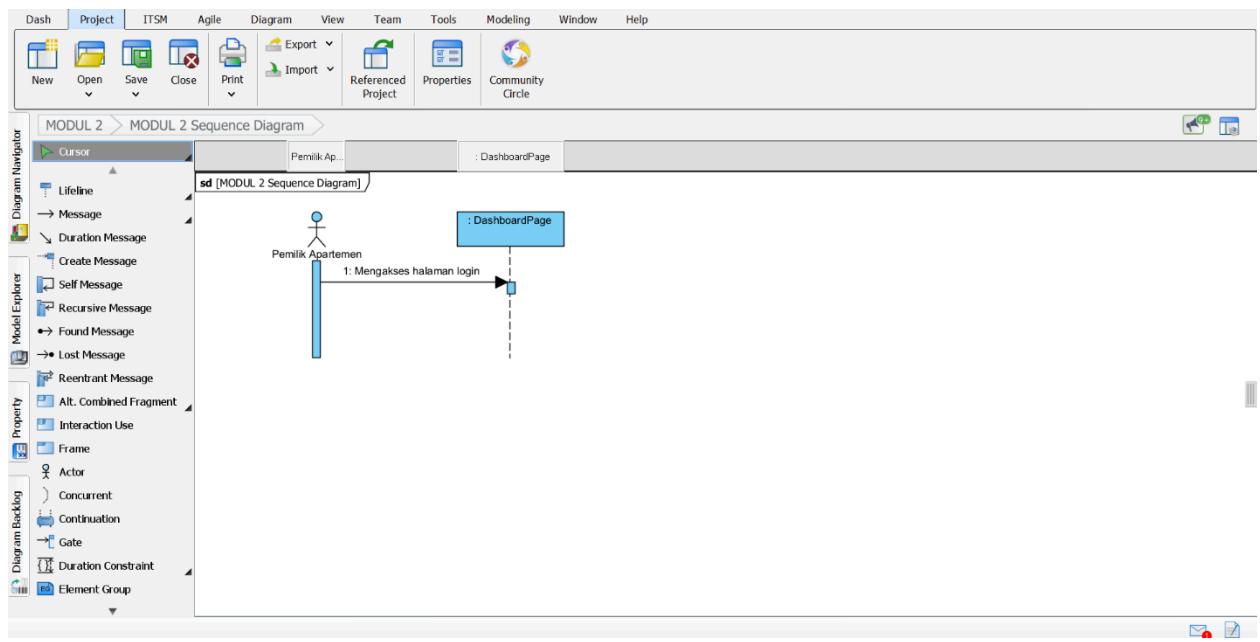
13. Tambahkan Lifeline dengan melakukan drag and drop dari toolbar ke worksheet. Untuk mengganti nama lifeline, double-click nama lifeline dan ganti namanya sesuai skenario studi kasus.



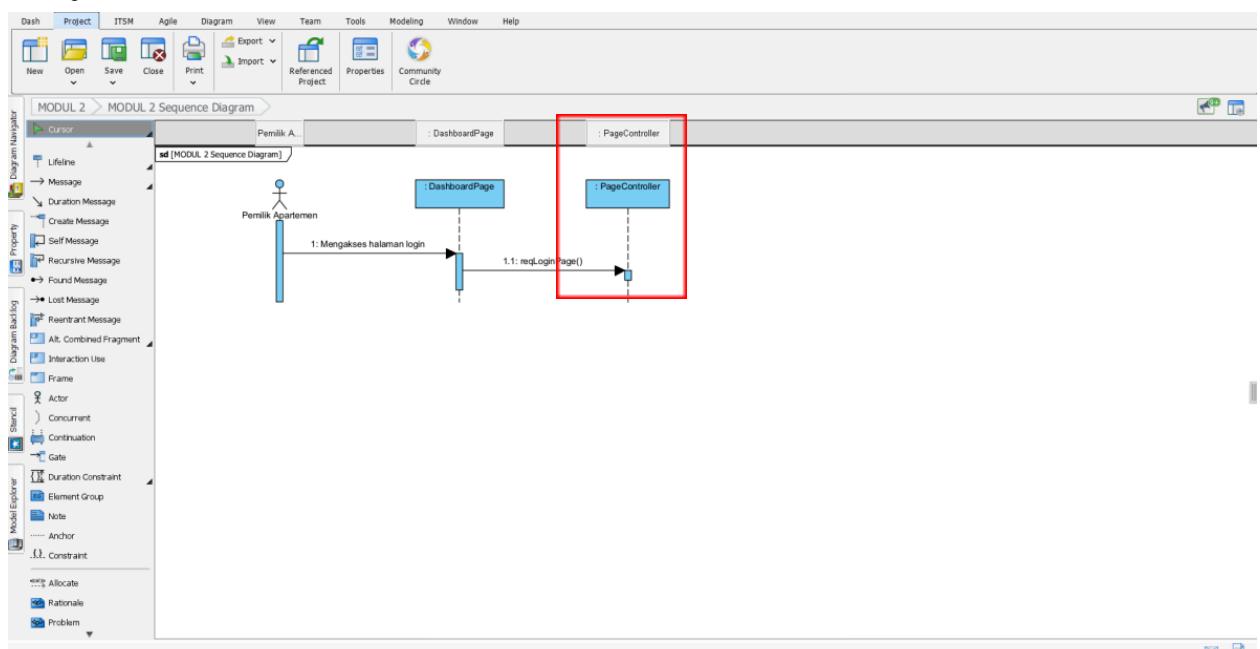


14. Klik *Message* pada toolbar, lalu lakukan *drag and drop* dari Actor ke *lifeline* yang dituju, lalu tambahkan nama pada *message* sesuai studi kasus.

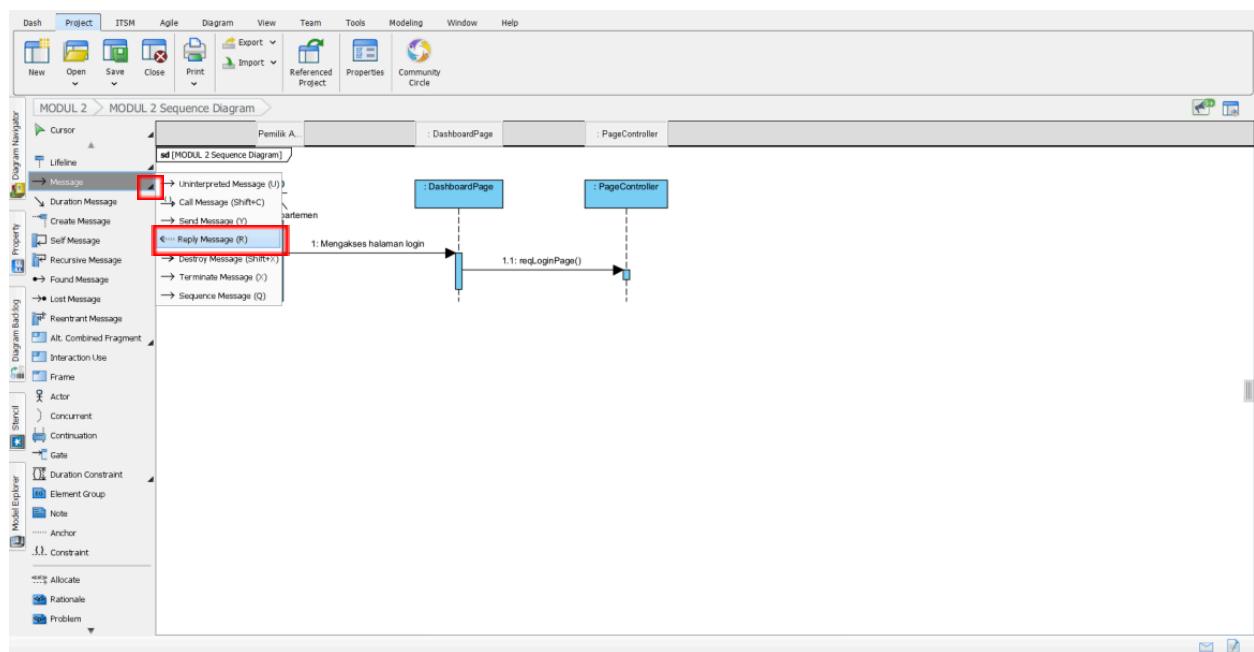




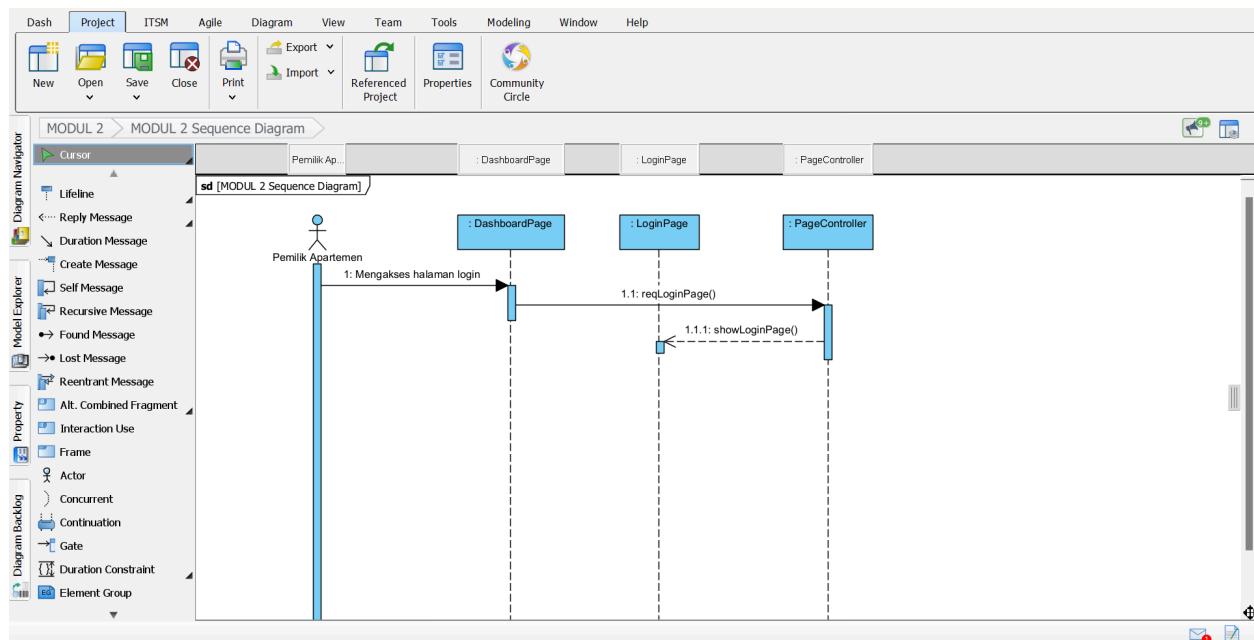
15. Tambahkan Lifeline Controller sebagai objek mediasi antara Boundary dan Entity.



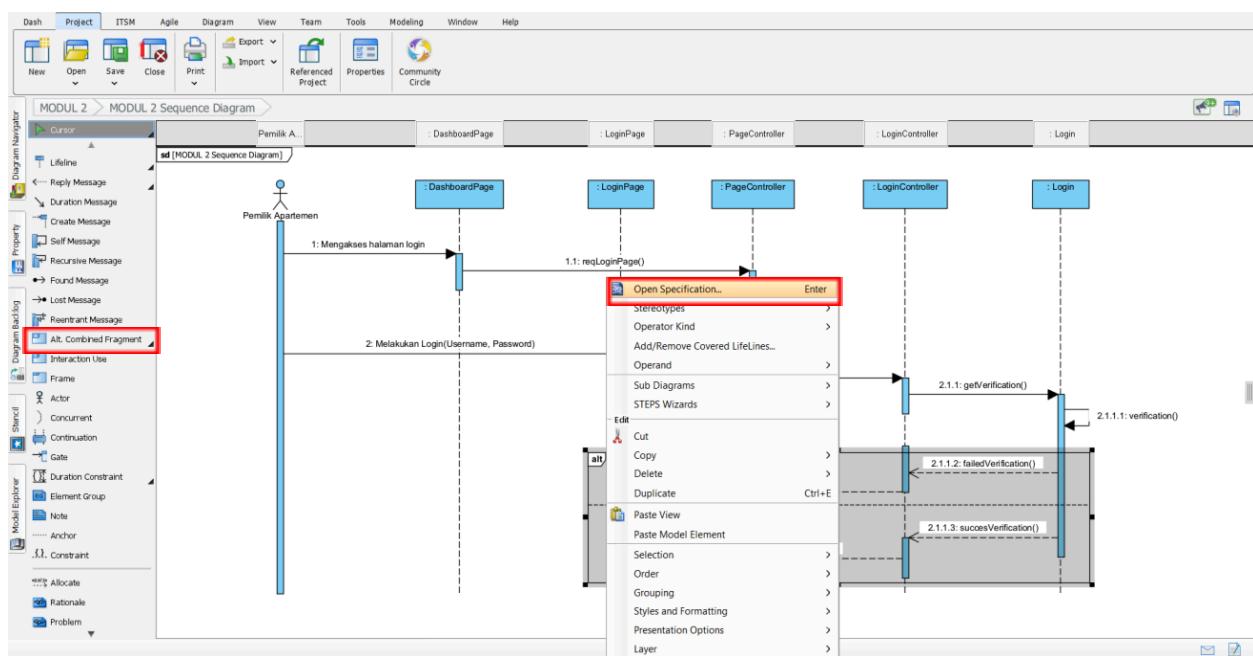
16. Tambahkan *Reply Message* untuk menambahkan timbal balik dari pemanggilan *method* dengan cara klik tanda segitiga siku-siku yang berada pada ujung *message* yang berada pada *toolbar* dan lakukan *drag and drop* notasi *replay message*.



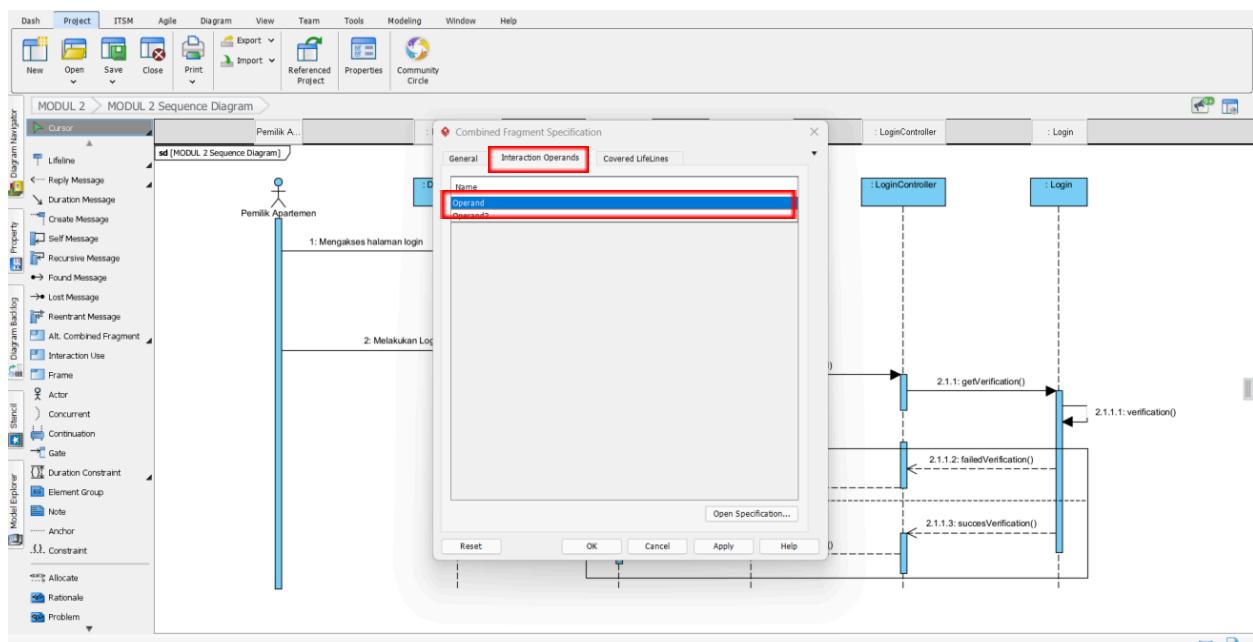
17. Lakukan *drag and drop* notasi *reply message* yang ada pada *toolbar* untuk menambahkan notasi yang diperlukan pada *sequence diagram*.



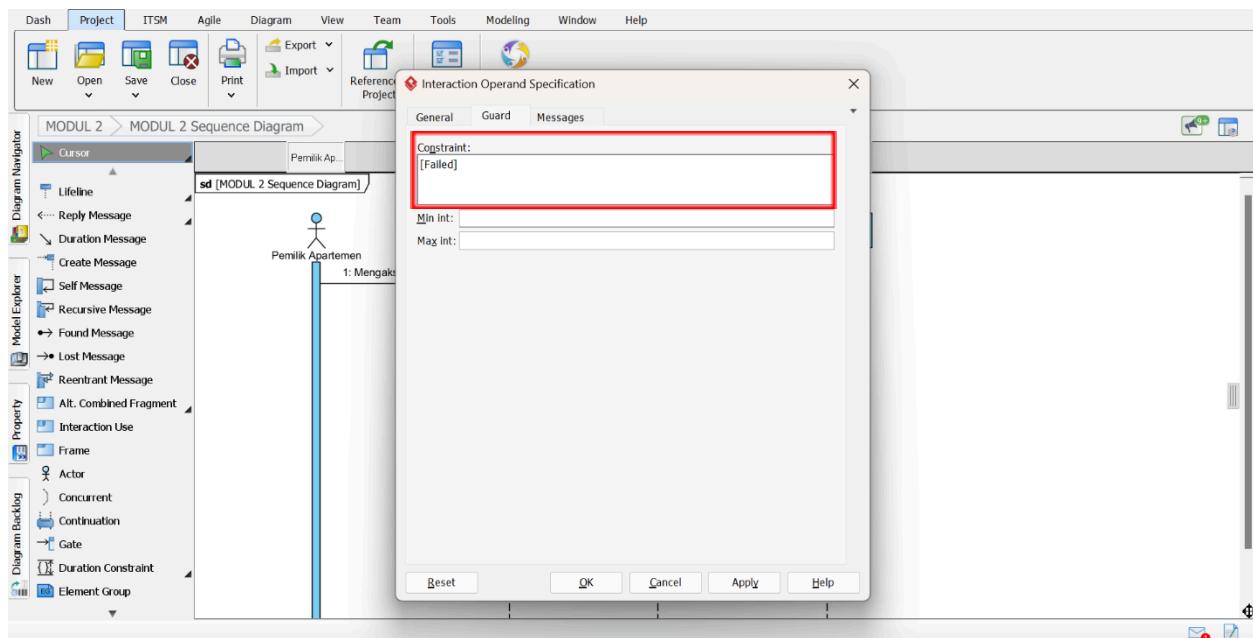
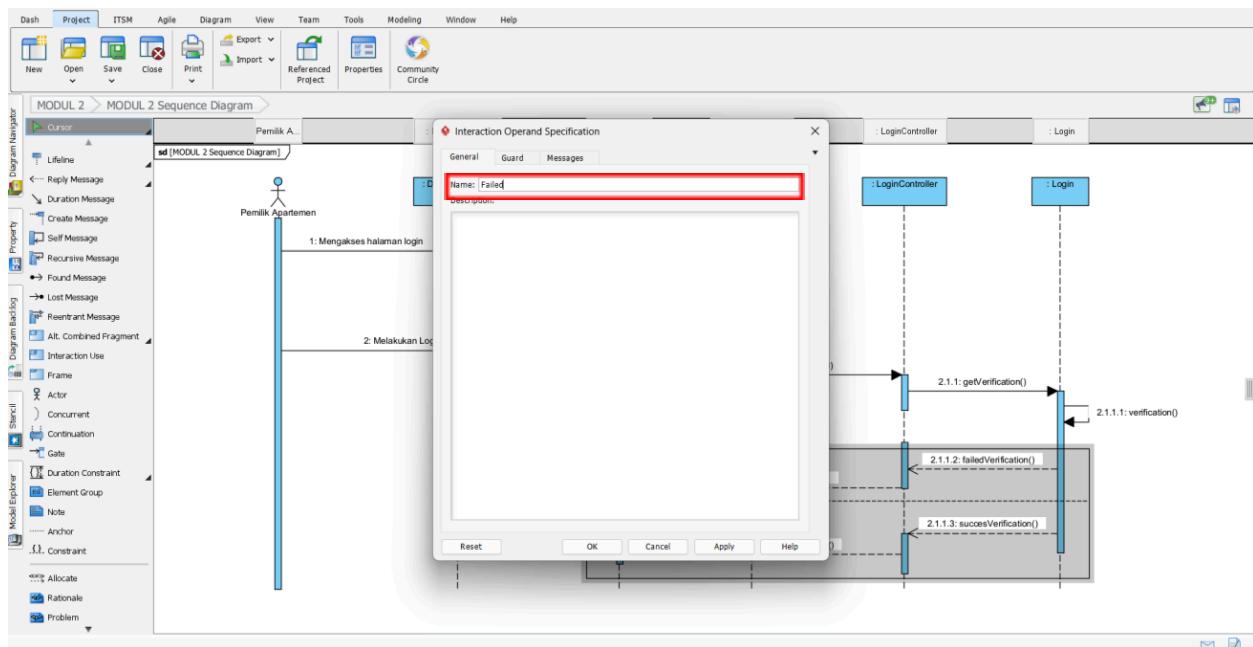
18. Jika ingin menambahkan frame Alt. Combined Fragment, lakukan drag and drop untuk menambahkan notasi pada toolbar. Tambahkan kondisi dengan melakukan klik kanan pada notasi Alt. Combined Fragment dan klik Open Specification.



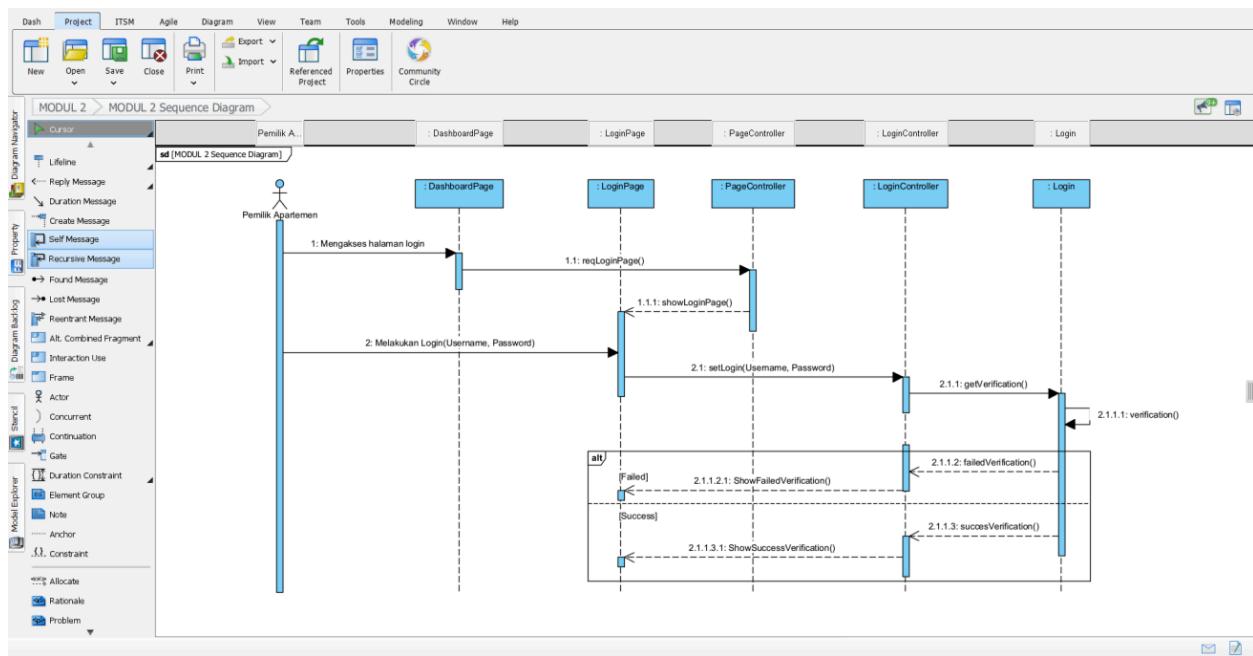
19. Klik dua kali Operand pada tab Interaction Operands.



20. Ganti Name pada tab General dan Constraint pada tab Guard sesuai dengan kondisi studi kasus. Lalu klik OK. Lakukan langkah yang sama untuk Operand2. Setelah itu klik OK untuk menyelesaikan pengaturan.



21. Berikut adalah tampilan hasil dari pengaturan Open Specification yang telah dilakukan pada notasi frame Alt. Combined Fragment.



CONTOH

A. Studi Kasus 1 (Level Mudah)

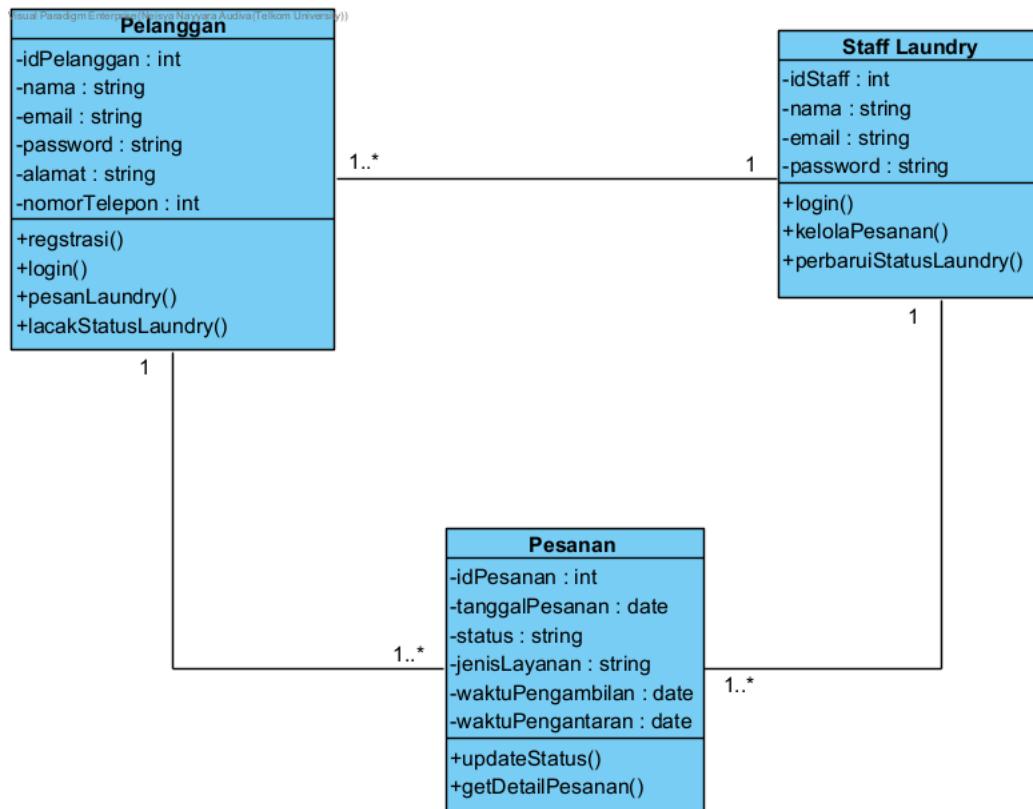
EAD Laundry adalah sebuah usaha laundry yang dimiliki oleh seorang pengusaha independen. Usaha ini menyediakan layanan laundry untuk masyarakat umum, termasuk mahasiswa, pekerja kantoran, dan keluarga. EAD Laundry dikenal karena layanan yang cepat, harga yang terjangkau, dan kualitas yang baik.

Seiring dengan meningkatnya permintaan, EAD Laundry berencana untuk mengembangkan sistem manajemen layanan laundry yang lebih efisien. Sistem ini akan memanfaatkan teknologi informasi untuk mempermudah proses pemesanan, pengelolaan, dan pelacakan status laundry.

Sistem yang akan dibangun adalah aplikasi berbasis web yang dapat diakses oleh pelanggan dan staf laundry. Sistem EAD Laundry dirancang untuk memberikan kemudahan dan kenyamanan bagi pelanggan dalam menggunakan layanan laundry. Salah satu fitur utama dari sistem ini adalah pendaftaran pelanggan, di mana pelanggan baru dapat dengan mudah membuat akun untuk mengakses berbagai layanan yang ditawarkan. Setelah mendaftar, pelanggan dapat melakukan pemesanan laundry secara online dengan memilih jenis layanan yang diinginkan, seperti cuci biasa, cuci kering, atau setrika, serta menjadwalkan waktu pengambilan yang sesuai dengan kebutuhan mereka.

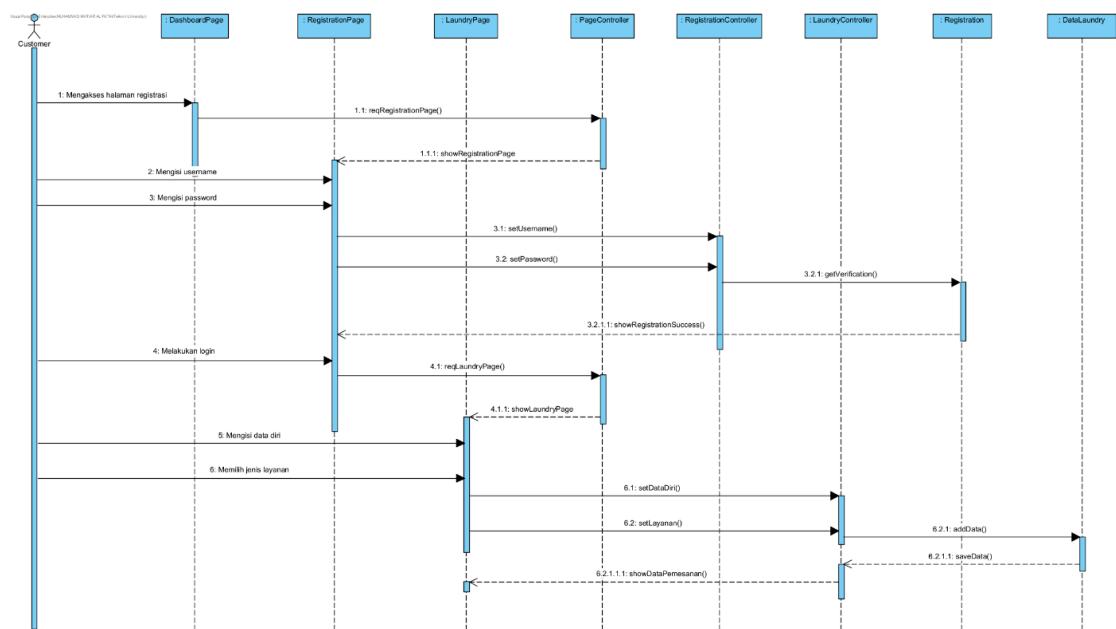
Setelah pemesanan dilakukan, pelanggan dapat melacak status laundry mereka melalui aplikasi, sehingga mereka selalu mendapatkan informasi terkini mengenai proses laundry yang sedang berlangsung. Fitur ini sangat membantu pelanggan untuk mengetahui kapan laundry mereka akan selesai dan siap untuk diambil atau diantar. Selain itu, sistem ini juga dilengkapi dengan manajemen staf yang memungkinkan. Dengan semua fitur ini, EAD Laundry berkomitmen untuk memberikan layanan yang cepat, efisien, dan berkualitas tinggi kepada semua pelanggan.

1. Class Diagram



2. Sequence Diagram (Diagram Sequence dibuat mengikuti Diagram Activity pada Modul 1)

- Berikut contoh Sequence Diagram pada fitur Pemesanan Laundry



B. Studi Kasus (Level Kompleks)

EAD Airlines merupakan sebuah perusahaan yang bergerak di bidang aviasi penerbangan. Perusahaan ini menyediakan berbagai layanan penerbangan, mulai dari pemesanan tiket, pembayaran tiket, dan check-in online oleh penumpang. Dalam beberapa tahun belakangan, EAD Airlines mengalami lonjakan permintaan dalam pemesanan tiket pesawat. Oleh karena itu, EAD Airlines ingin membangun sebuah sistem yang terintegrasi satu sama lain guna membangun efisiensi dalam setiap proses bisnis yang ada di dalam EAD Airlines. Sistem ini harus dapat digunakan oleh pihak yang berkepentingan di EAD Airlines, seperti Admin website dan penumpang.

Berdasarkan kebutuhan yang sudah dijelaskan diatas, untuk merancang sistem diperlukan class diagram. Class-class yang harus ada, diantaranya Penumpang, Pemesanan, Penerbangan, E-ticket, dan Admin. Setiap class ini memiliki relasi satu sama lainnya. Class e-ticket memiliki ketergantungan yang kuat dengan class Pemesanan, dimana jika class Pemesanan tidak ada, maka class E-ticket juga tidak ada.

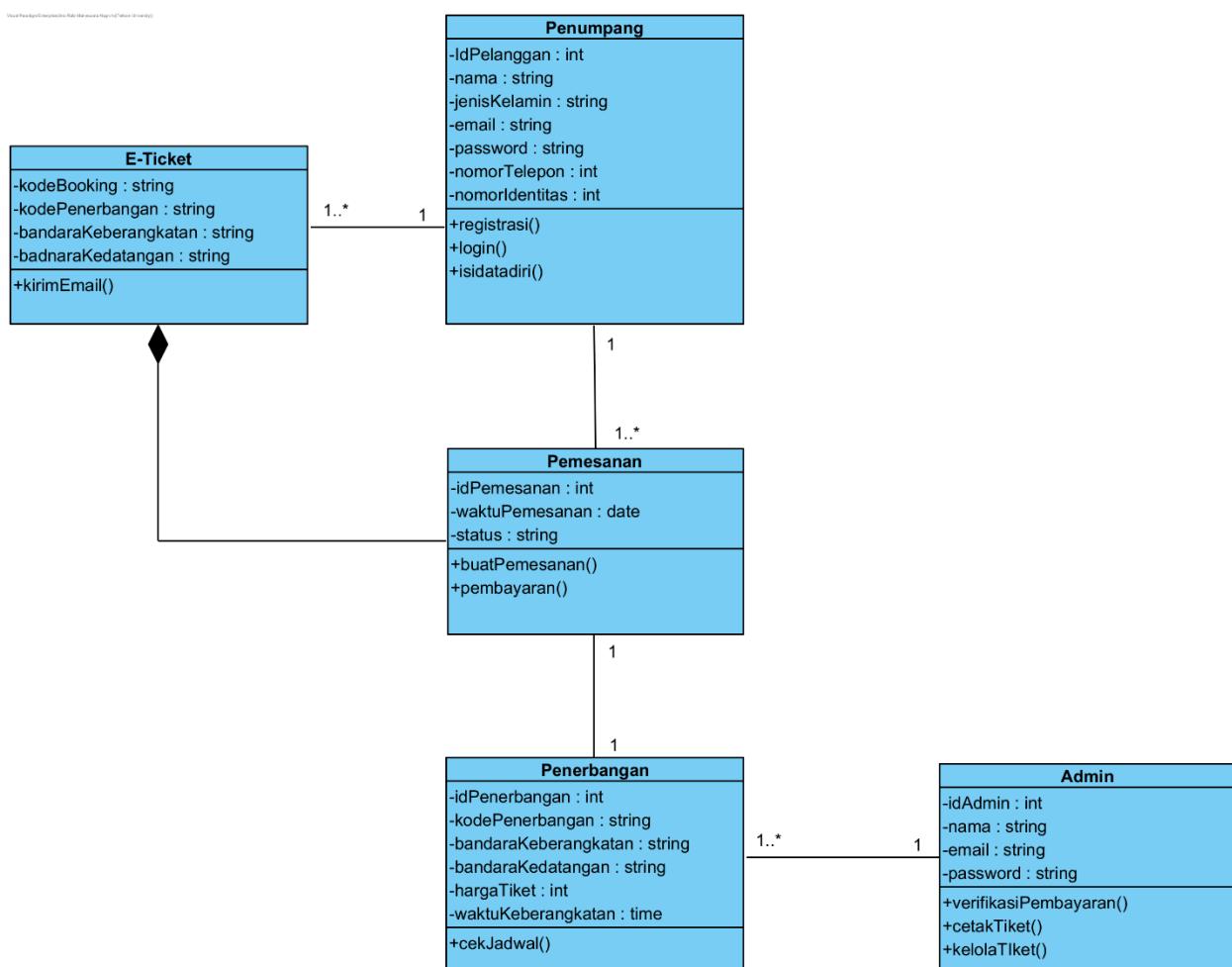
Sebelum mengakses layanan yang ada di dalam sistem EAD Airlines, penumpang harus mendaftarkan dirinya. Setelah melakukan registrasi, penumpang dapat mengakses halaman login sesuai dengan akun yang teregistrasi pada sistem. Selanjutnya halaman web akan mengarahkan ke halaman pengisian data diri, seperti nama lengkap, jenis kelamin, dan nomor identitas penumpang. Setelah mengisi data diri, penumpang dapat melakukan pemesanan tiket pada website EAD Airlines.

Penumpang dapat memilih layanan pemesanan di dalam website EAD Airlines. Pada halaman ini, pasien diarahkan untuk memilih bandara keberangkatan (departure), bandara kedatangan (arrival), dan waktu keberangkatan. Selanjutnya, sistem akan menampilkan daftar penerbangan lengkap dengan harga tiketnya. Penumpang dapat memilih tiket yang tersedia sesuai dengan preferensinya. Setelah memilih tiket, penumpang dapat melakukan pembayaran. Setelah pembayaran sukses, maka sistem akan mencetak e-ticket dan dikirimkan melalui email penumpang yang terdaftar. Setelah penumpang mendapatkan e-ticket penerbangannya, ia dapat melakukan check-in online dengan memasukkan kode booking yang terdapat di dalam e-ticket penerbangan. Setiap penumpang memiliki kode booking yang

berbeda-beda. Setelah melakukan pemesanan penumpang dapat melihat riwayat pemelian.

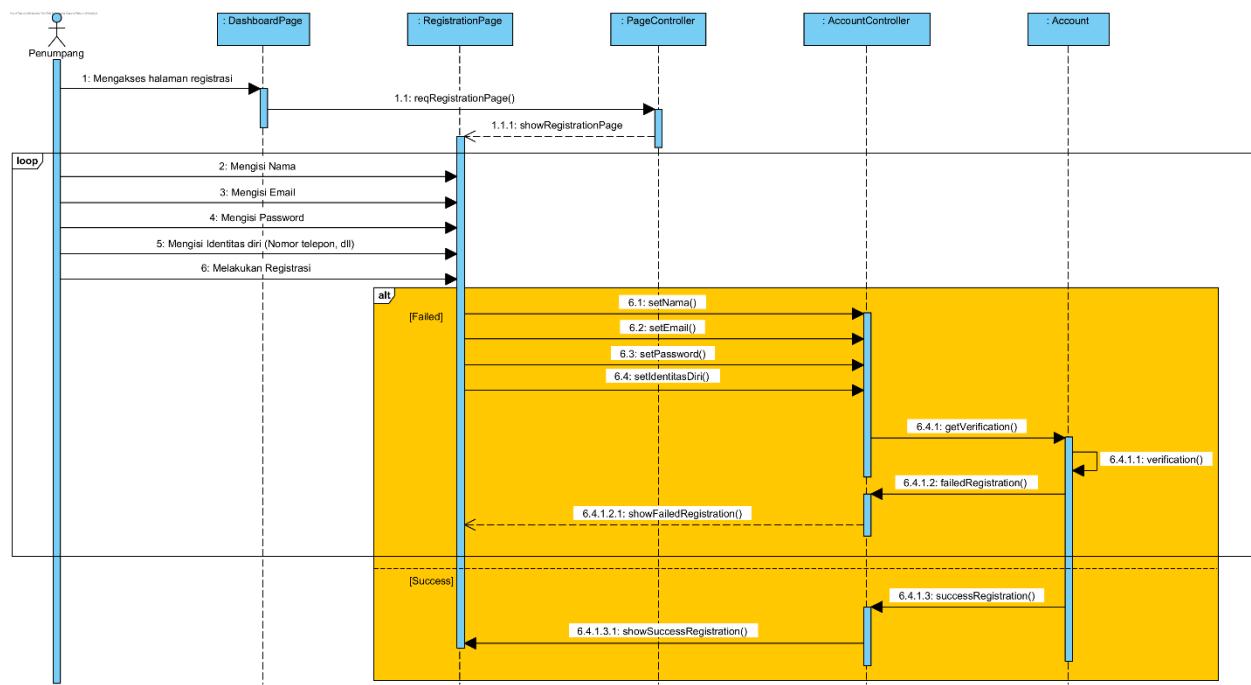
Sementara itu, admin sistem EAD Airlines memiliki peran dalam memastikan kelancaran operasional terkait transaksi tiket pesawat. Tugas utamanya meliputi verifikasi setiap pembayaran yang dilakukan oleh penumpang sebelum tiket diterbitkan. Setelah pembayaran diverifikasi, admin bertanggung jawab mencetak e-ticket yang akan digunakan oleh penumpang sebagai bukti pemesanan resmi. Admin juga mengelola daftar tiket pesawat pada sistem EAD Airlins. Mereka dapat menambah, mengedit, atau menghapus tiket yang tersedia, menyesuaikan harga tiket berdasarkan kebijakan EAD Airlines, serta memastikan bahwa data penerbangan yang terdaftar selalu diperbarui sesuai dengan jadwal maskapai.

1. Class Diagram



2. Sequence Diagram

- Berikut contoh Sequence Diagram pada fitur Registrasi



“

Sebagai Shinobi Konoha

Kita Harus Punya Sifat

Pantang Menyerah!

”