Nama : Theresia Agatha Napitupulu
NIM : 12030123120010
Mata Kuliah : Analisis dan Desain Sistem (D)

Unified Modeling Language

A. Pengertian UML

Unified Modeling Language (UML) adalah sebuah standar bahasa visual yang digunakan untuk memodelkan sistem perangkat lunak. UML membantu pengembang, analis, dan desainer perangkat lunak dalam mendokumentasikan, merancang, dan memahami struktur serta perilaku sistem dengan lebih baik. Dikembangkan oleh *Object Management Group* (OMG), UML sangat fleksibel karena dapat digunakan untuk berbagai jenis sistem, baik perangkat lunak maupun non-perangkat lunak.

B. Manfaat UML

- Komunikasi yang Lebih Efektif: UML mempermudah tim pengembang, analis bisnis, dan pemangku kepentingan dalam berkomunikasi melalui diagram yang jelas dan mudah dipahami.
- Dokumentasi yang Konsisten: UML menawarkan format dokumentasi yang konsisten, memungkinkan semua pihak yang terlibat dalam proyek untuk memahami desain sistem tanpa perlu mempelajari metode yang berbeda.
- Pemecahan Masalah yang Lebih Mudah: Dengan UML, pemodelan sistem dilakukan secara visual, membuatnya lebih mudah untuk mengidentifikasi dan memperbaiki potensi masalah dalam desain sebelum implementasi sistem.
- Mendukung Perencanaan dan Pengembangan: UML membantu merinci kebutuhan sistem, mempermudah pengembangan, pemeliharaan, dan peningkatan sistem di masa depan.

C. Jenis Diagram dalam UML

UML menggunakan elemen-elemen dan bentuk association antara UML dengan diagramnya. Diagram UML diklasifikasikan menjadi dua, yaitu:

1. Structural Diagram

Structural diagram membantu dalam menggambarkan statis aspek atau struktur dari sebuah sistem. *Structural diagram* meliputi: *Component Diagram, Object Diagram, Class Diagram, dan Deployment Diagram.*

a) Component Diagram

Component diagram sering digunakan untuk merepresentasikan bagaimana komponen-komponen fisik di dalam sistem telah terorganisir. *Component diagram* menggambarkan hubungan struktural antara elemen sistem perangkat lunak dan membantu memahami jika persyaratan fungsional telah dicakup oleh pengembangan yang direncanakan.

b) Class Diagram

Class diagram adalah diagram UML yang paling banyak digunakan. *Class diagram* merupakan building blocks dari segala orientasi objek pada sistem perangkat lunak. *Class diagram* dapat digunakan untuk menggambarkan struktur statis dari sebuah sistem dengan menunjukkan kelas-kelas sistem, metodenya, dan atribut. *Class diagram* bisa membantu Anda untuk mengidentifikasi hubungan antara kelas-kelas yang berbeda maupun objek-objek yang berbeda.

c) Object Diagram

Object diagram membantu dalam menggambarkan perilaku (behaviour) ketika objek telah dipakai. *Object diagram* juga membantu menggambarkan pengklasifikasi aktual dan hubungannya menggunakan *class diagram.*

d) Composite Structure Diagram

Composite structure diagram merepresentasikan hubungan antara bagian-bagian dan konfigurasinya yang menentukan bagaimana *classifier (class, komponen, atau deployment code)* berperilaku.

e) Deployment Diagram

Deployment diagram digunakan untuk merepresentasikan perangkat keras sistem dan perangkat lunaknya. *Deployment diagram* akan menjelaskan terkait komponen perangkat keras apa dan komponen perangkat lunak apa yang berjalan di dalamnya. *Deployment diagram* mengilustrasikan arsitektur sistem sebagai distribusi artefak perangkat lunak di atas target yang terdistribusi.

f) Package Diagram

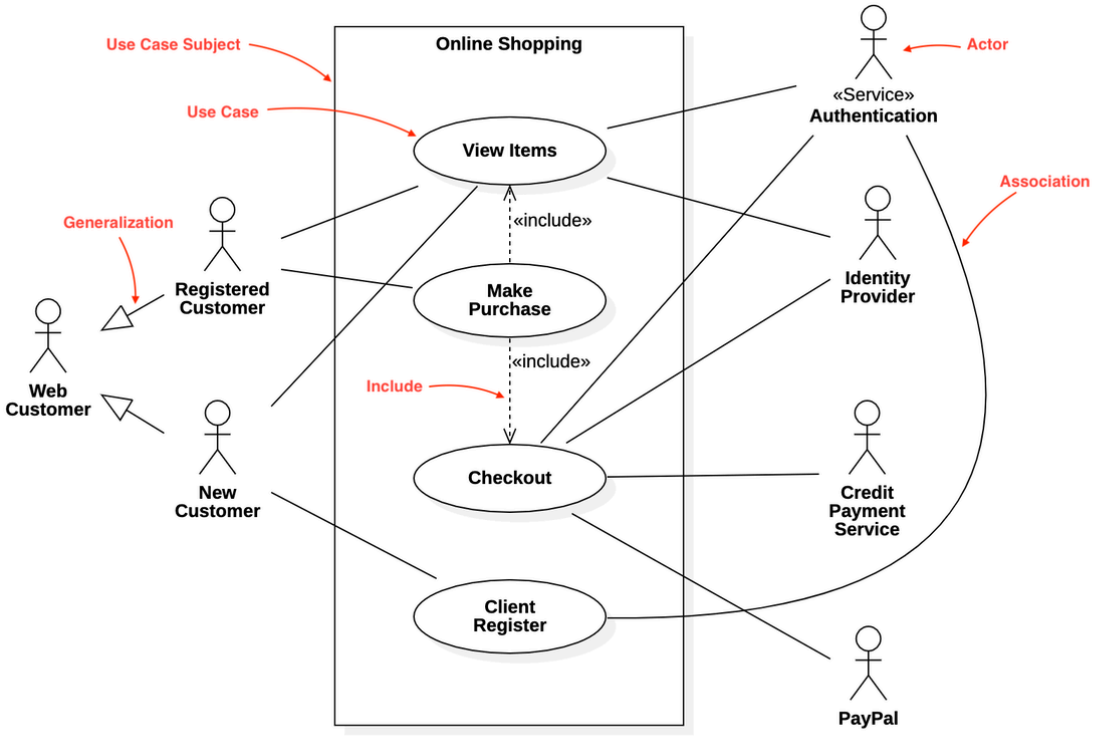
Secara sederhana, *package* diagram menunjukkan dependensi antara *package* yang berbeda dan komposisi internal dari package. *Package* dapat membantu untuk menata diagram UML menjadi kelompok-kelompok yang berarti dan membuat diagram untuk mudah dipahami. Pada dasarnya *package* diagram digunakan untuk menata class diagram dan use case diagram.

2. Behavioral Diagram

Behavioral diagram membantu dalam menggambarkan aspek dinamis atau perilaku dari sebuah sistem. *Behavioral diagram* meliputi: *Use Case Diagram, State Diagram, Activity Diagram, dan Interaction Diagram.*

a) Use Case Diagram

Use case diagram digunakan untuk menggambarkan fungsionalitas dari sistem dan bagian dari sistem. *Use case diagram* digunakan secara luas untuk mengilustrasikan kebutuhan fungsional dari sistem dan interaksinya dengan agen eksternal *(actors*).

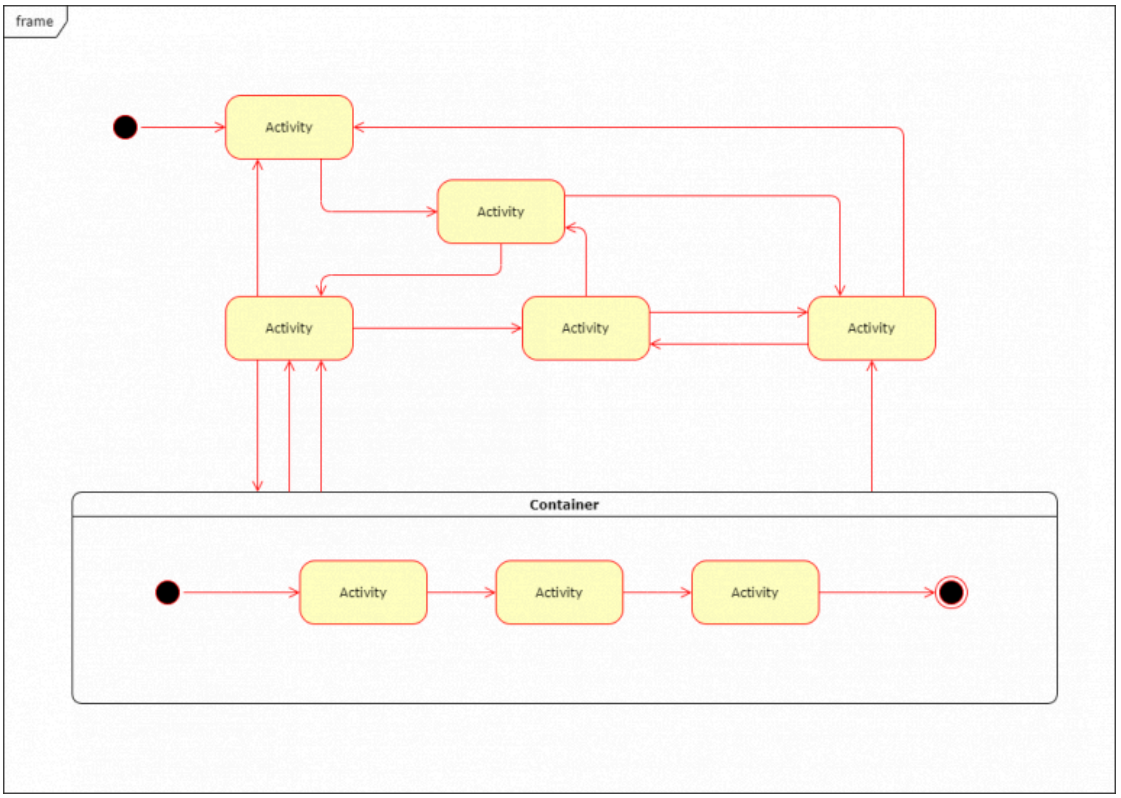


Use Case Diagram (Foto: docs.staruml.io)

b) *State Machine Diagram*

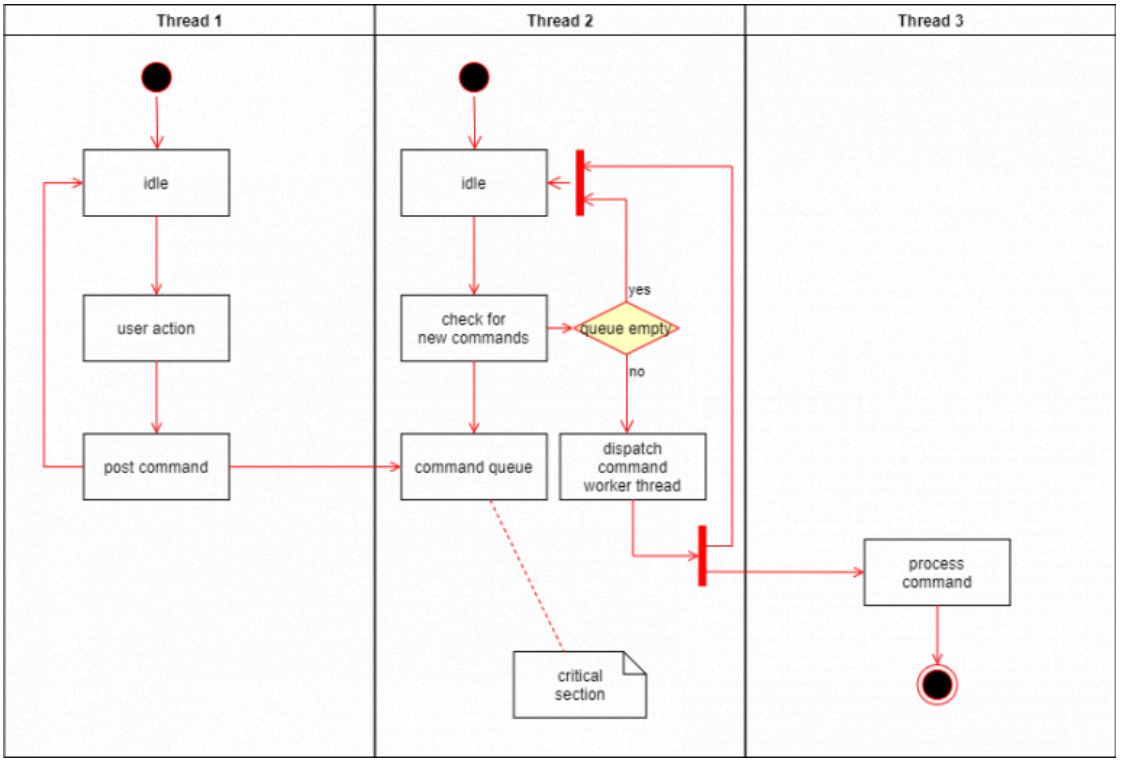
State machine diagram digunakan untuk merepresentasikan kondisi dari sistem atau bagian dari sistem pada waktu yang terbatas. Jenis *behavioral diagram* ini juga sering disebut *state machine* atau *state-chart diagram*. Jenis ini digunakan untuk

memodelkan perilaku yang dinamis dari sebuah *class* sebagai bentuk respons waktu dan mengubah *external stimuli*.

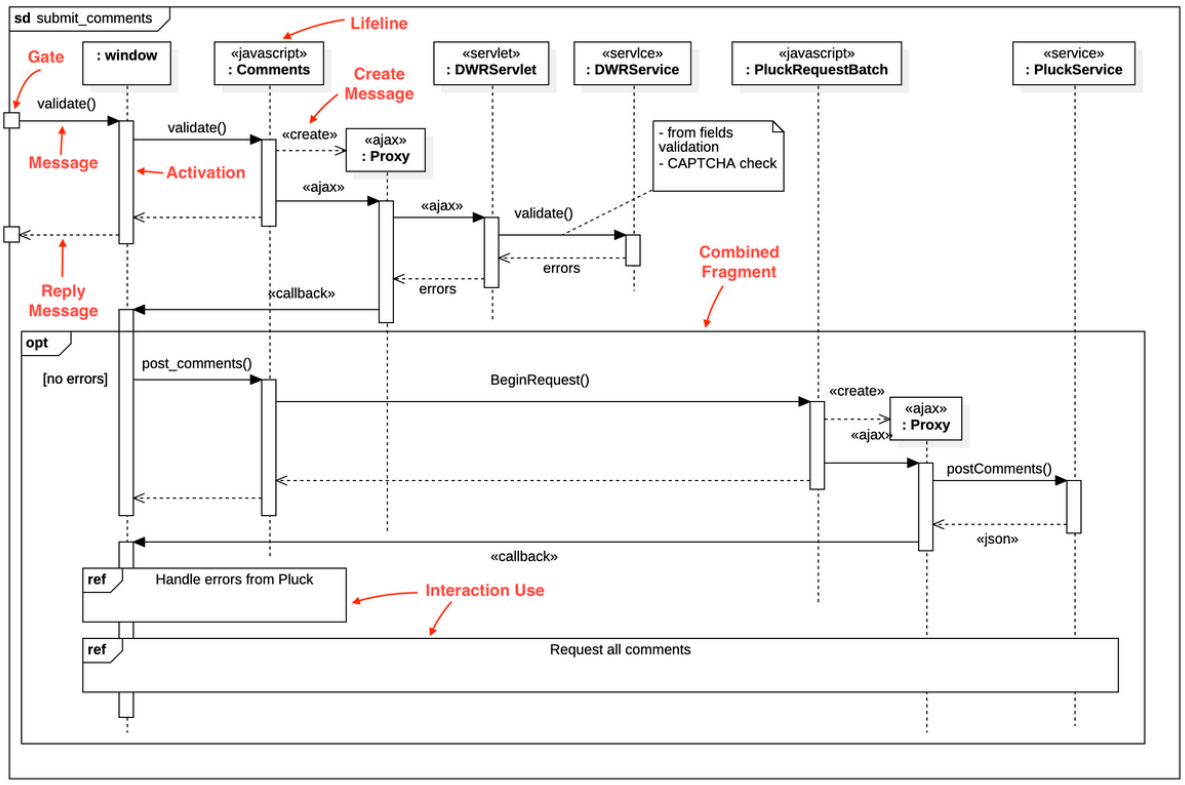


c) *Activity Diagram*

activity diagram digunakan untuk mengilustrasikan *flow control* dalam sebuah sistem. *Activity diagram* berfokus pada kondisi bagaimana *flow* dan dan *sequence* dalam waktu terjadinya. *Activity diagram* digunakan untuk menggambarkan atau mendeskripsikan penyebab dari beberapa kejadian.



Sequence diagram merupakan diagram yang menjelaskan interaksi objek berdasarkan urutan waktu. *Sequence* dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu, seperti yang tertera pada Use Case diagram. Diagram ini sering digunakan oleh pebisnis dan *software developers* untuk mendokumentasikan dan memahami kebutuhan untuk sistem yang sudah ada dan sistem yang baru.



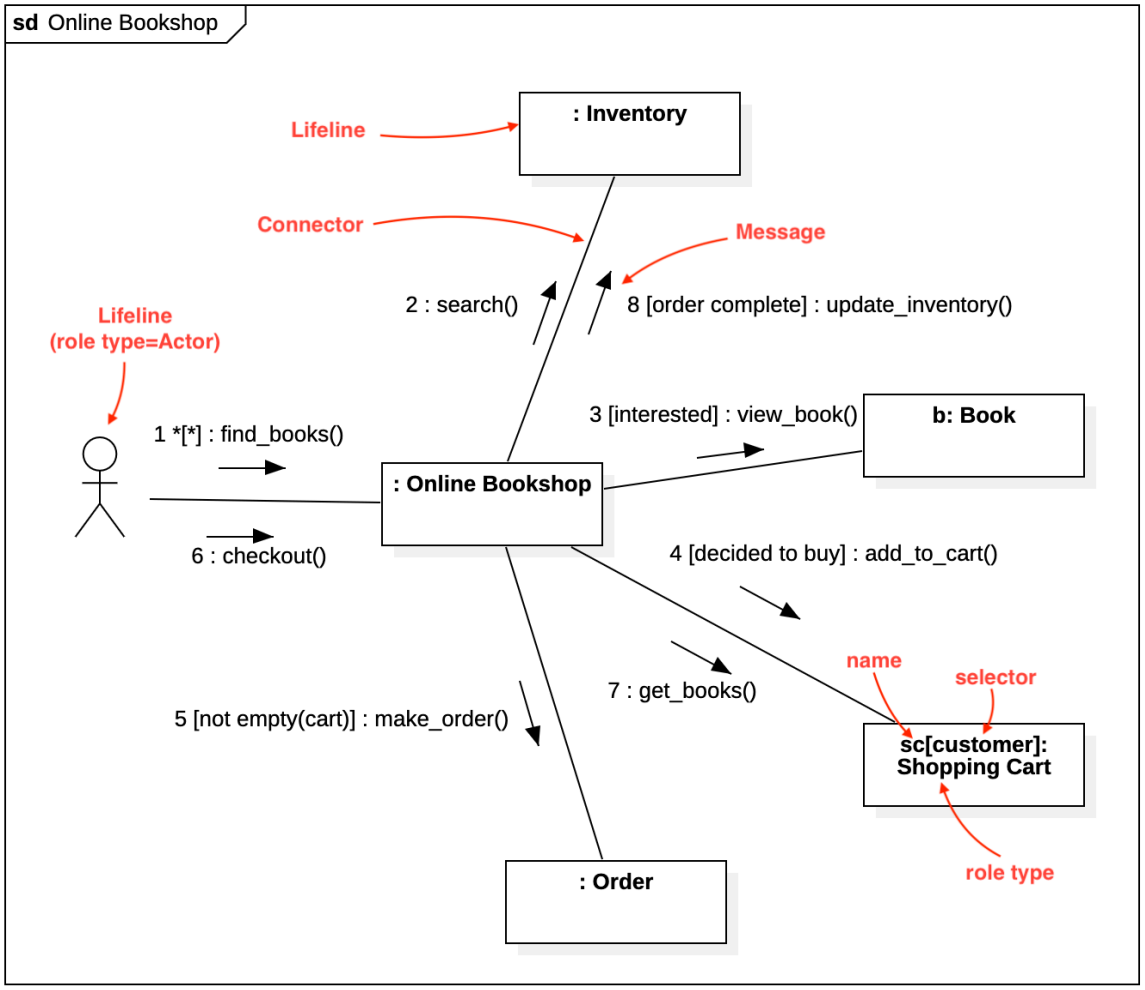
Sequence Diagram (Foto: docs.staruml.io)

e) *Timing Diagram*

Timing diagram digunakan untuk menggambarkan perilaku objek dalam batasan waktu tertentu.

f) *Communication Diagram*

Communication diagram, juga dikenal sebagai *collaboration diagram* pada UML 1.x, digunakan untuk menunjukkan pesan berurutan yang ditukarkan antara objek. Fokus utama dari *communication diagram* adalah objek dan hubungannya.



Communication Diagram (Foto: docs.staruml.io)

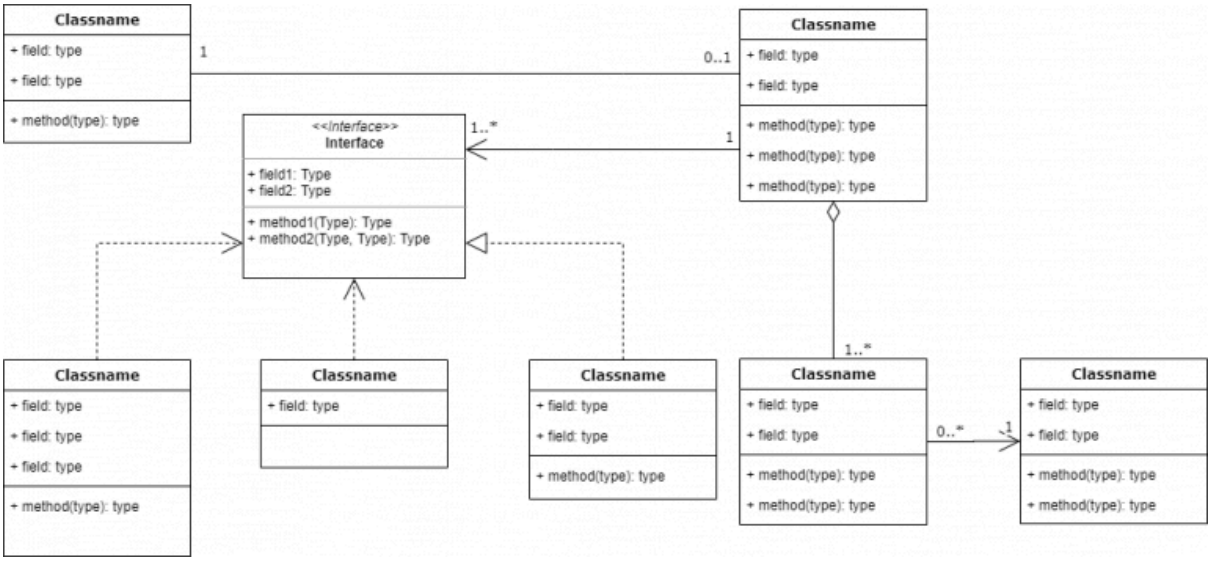
g) *Interaction Overview Diagram*

Sebuah *interaction overview diagram* memodelkan sebuah aksi yang berurutan dan membantu menyederhanakan interaksi yang kompleks menjadi kejadian yang lebih sederhana. *Interaction overview diagram* adalah percampuran dari *activity diagram* dan *sequence diagram*.

h) *Class Diagram*

Digunakan untuk merepresentasikan struktur sistem, termasuk kelas, atribut, dan metode yang digunakan oleh kelas tersebut. Menunjukkan hubungan antara kelas-kelas, seperti pewarisan

dan asosiasi. Membantu para pengembang untuk memodelkan struktur dan hubungan antara kelas dalam sistem.



Langkah Membuat UML

Berikut adalah langkah-langkah dasar dalam membuat *Unified Modeling Language*:

1. Tentukan Tujuan dan Scope: Identifikasi komponen sistem mana yang perlu dipetakan. Ini bisa termasuk kelas, objek, komponen fisik, dan interaksi antar elemen.
2. Pilih Jenis Diagram yang Sesuai: Sesuaikan diagram UML dengan aspek sistem yang ingin digambarkan. Misalnya, gunakan *class diagram* untuk hubungan antar kelas, dan *sequence diagram* untuk alur komunikasi.
3. Buat Diagram: Gunakan alat bantu seperti *software* pemodelan (misalnya, Lucidchart atau Visual Paradigm) untuk membuat diagram UML yang diinginkan.
4. Verifikasi dan Validasi: Pastikan diagram sudah sesuai dengan kebutuhan sistem dan mudah dipahami oleh semua pemangku kepentingan.

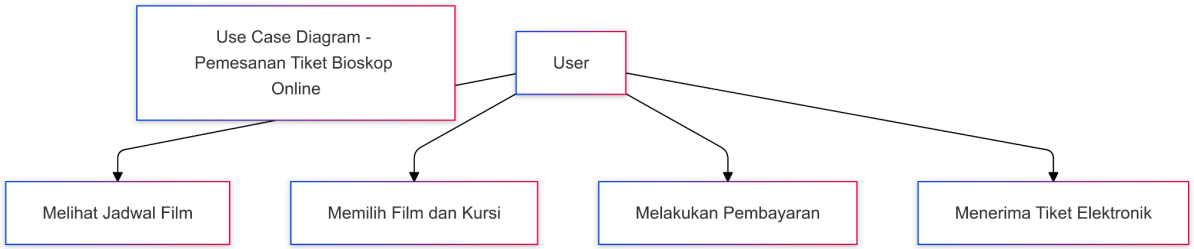
Studi Kasus UML

Kasus: Sistem Pemesanan Tiket Bioskop Online

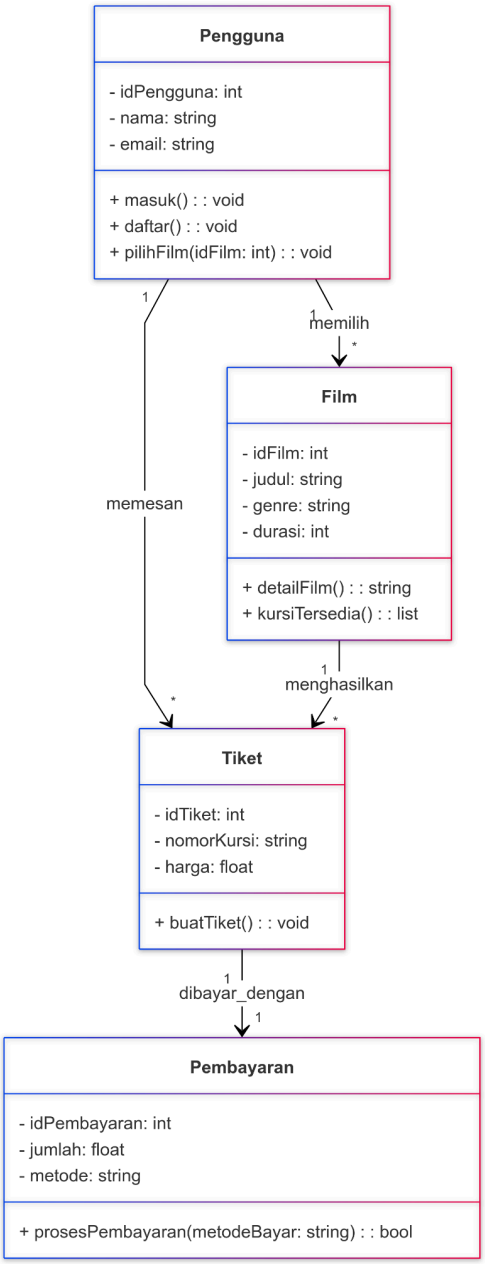
Deskripsi Sistem:

Sebuah aplikasi menyediakan layanan pemesanan tiket bioskop secara online. Pengguna dapat melihat jadwal film, memilih film, kursi, dan waktu pemutaran, lalu melakukan pembayaran. Sistem akan memberikan tiket elektronik kepada pengguna setelah pembayaran berhasil.

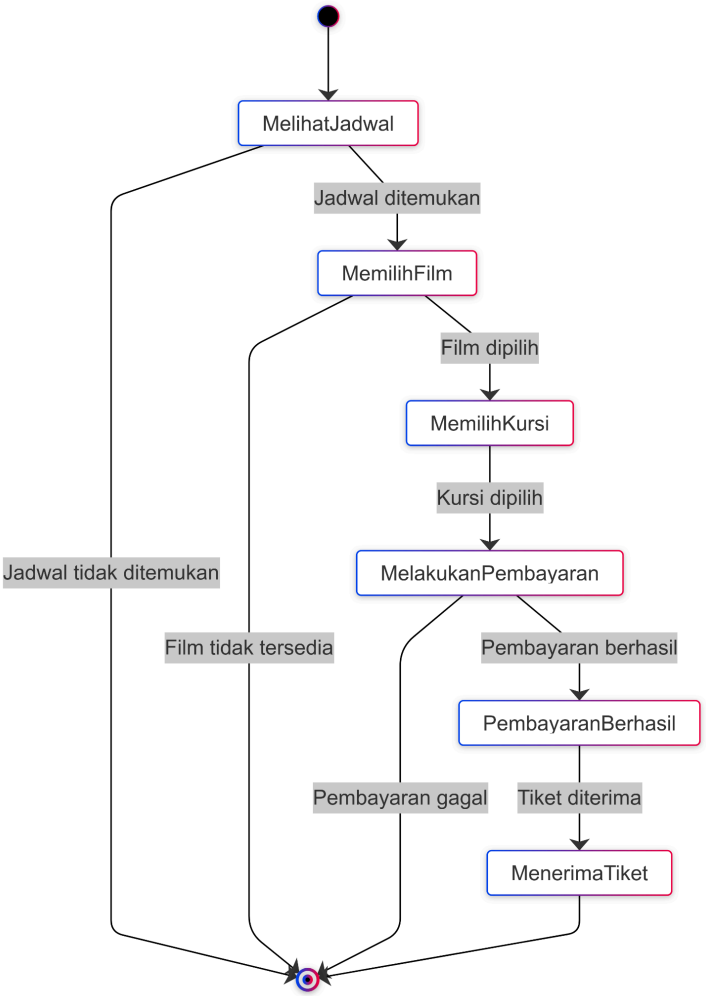
Use Case Diagram



Class Diagram



Activity Diagram



Kesimpulannya

UML adalah alat penting dalam rekayasa perangkat lunak modern, karena memberikan panduan visual yang jelas untuk memahami, merancang, dan membangun sistem kompleks. Dengan menguasai berbagai jenis diagram UML, pengembang dapat menciptakan sistem yang sesuai kebutuhan pengguna, terstruktur dengan baik, dan mudah dipelihara.