

UML

Unified Modelling Language

Birleşik Modelleme Dili

Ders notu

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/> ve
<https://online.visual-paradigm.com/diagrams/tutorials/>

linklerinden oluşturulmuştur.

UML Diyagramında use case diyagramının rolü?

use case diyagramı, bir aktörün rolü ile sistem arasındaki etkileşimlerini tanımlar.

use case diyagramı bir amaca ulaşmayı hedefleyen eylem /olay adımlarının bir listesidir.

use case diyagramı , sistem gereksinimlerini belirlemek ve organize etmek üzere kullanılır.

use case diyagramı, uygulanacak özellikler ve olası hataların çözümünü tanımlar; sistem ve kullanıcılar arasındaki olası etkileşim dizilerinden oluşur.

UML Diyagramında use case diyagramının rolü?

Bir use case dşyagramı ya da use case diyagramları kümesinin temel özellikleri:

Fonksiyonel gereksinimlerin görsel tasarımını gerçekleştirir.

Sistem ile aktörlerin (kullanıcıların)) etkileşimlerinin hedeflerini modeller.

Bir ana olayın akışını (ana senaryoları ve olası akışlar (alternatifleri) görsel çözümler.

Aktör, use case, İlişkiler (Relationship) , Sistem Sınırı (System Boundary)

- ❑ Aktörler genellikle rollerine göre tanımlanan, sistemle ilgili bireylerdir. Aktör bir insan veya başka bir harici sistem olabilir.
- ❑ use case , aktörlerin belirli bir amaca ulaşmak için bir sistemi nasıl kullandığını açıklar.
 - ❖ Use case ler hedefe ulaşmak üzere faaliyetleri ve bunların varyasyonlarını açıklayan hedefleri yerine getirmek üzere bir kullanıcı tarafından başlatılır.
- ❑ Aktörler ve kullanım durumları arasındaki ilişkiler belirlenir.
- ❑ Sistem sınırı, ilgili sistemi çevresindeki dünya ile ilişkili olarak tanımlar.

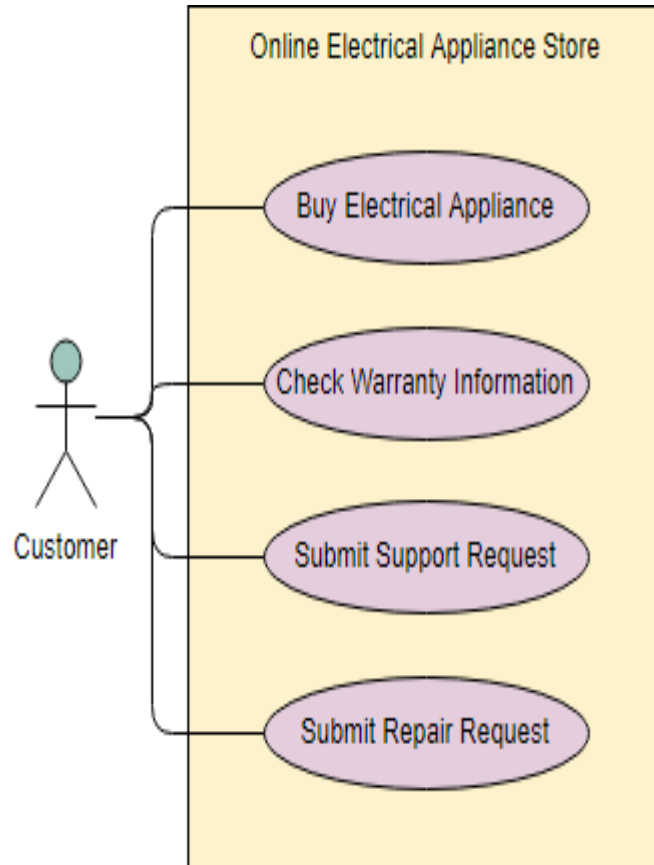
Niçin use case diyagramı?

- ❑ Kara kutu olarak adlandırılan formda fonksiyonel gereksinimler belirlenir ve belgelenir.
 - ❑ Doğal dilde yazıldığı için müşteri /kullanıcılarla iletişim kurmanın en etkili yoludur.
 - ❑ Problem ana kullanıcı özelliklerine (use case lere) bölünerek büyük projelerin karmaşıklığı yönetilir; aynı zamanda uygulama kullanıcı bakış açısından geliştirilir.
 - ❑ Bir sequence diyagramında yer alan bir use case, birden fazla nesne ve sınıfın birlikte nasıl işleme girdiğini gösterir.
 - ❖ Nesneleri ve sınıfları birbirine bağlayan mesajlar, metodlar (işlemler) ve verilerin parametreleriyle belirlenir.
 - ❑ Üst düzey modellerin (aktörler ve işbirliğindeki nesneleri arasındaki etkileşimlerin) doğrulanması (verification) ile daha sonra fonksiyonel gereksinimlerin (doğrulanması (validation)), yani beyaz kutu testinin planı) sağlanır.
- Bu yaklaşımla: Hedefler ve amaçlar kullanıcı bakış açısından belirlenir; uygulanan, testi yapılan ve teslim edilen use cases yazılım geliştirme süreçlerinin izlenebilirliğini (traceability) sağlar.

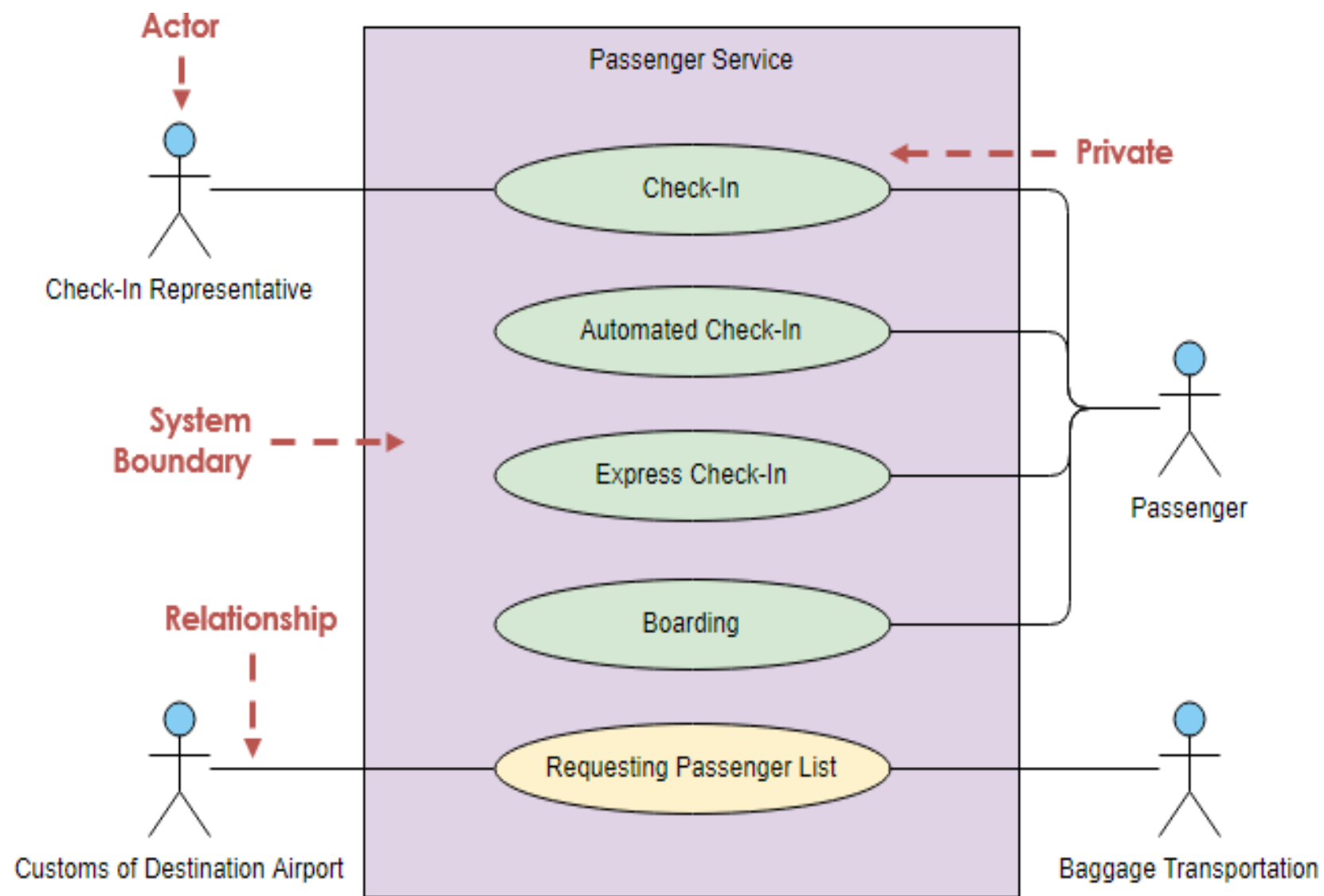
Use case Diyagramının Planlanması

- ❑ Sistemin aktörleri, yani kullanıcıların rolleri belirlenir.
- ❑ Her kullanıcı kategorisi için, sistemle ilgili olarak kullanıcıların gerçekleştirecekleri rollerin tümü belirlenir.
- ❑ Bu hedeflere ulaşmak için kullanıcıların sistemin hangi işlemleri gerçekleştirmesi gerektirdiği belirlenir.
- ❑ Her hedef için use case oluşturulur.
- ❑ use case ler yapılandırılır; yani ayrıntılandırılır.

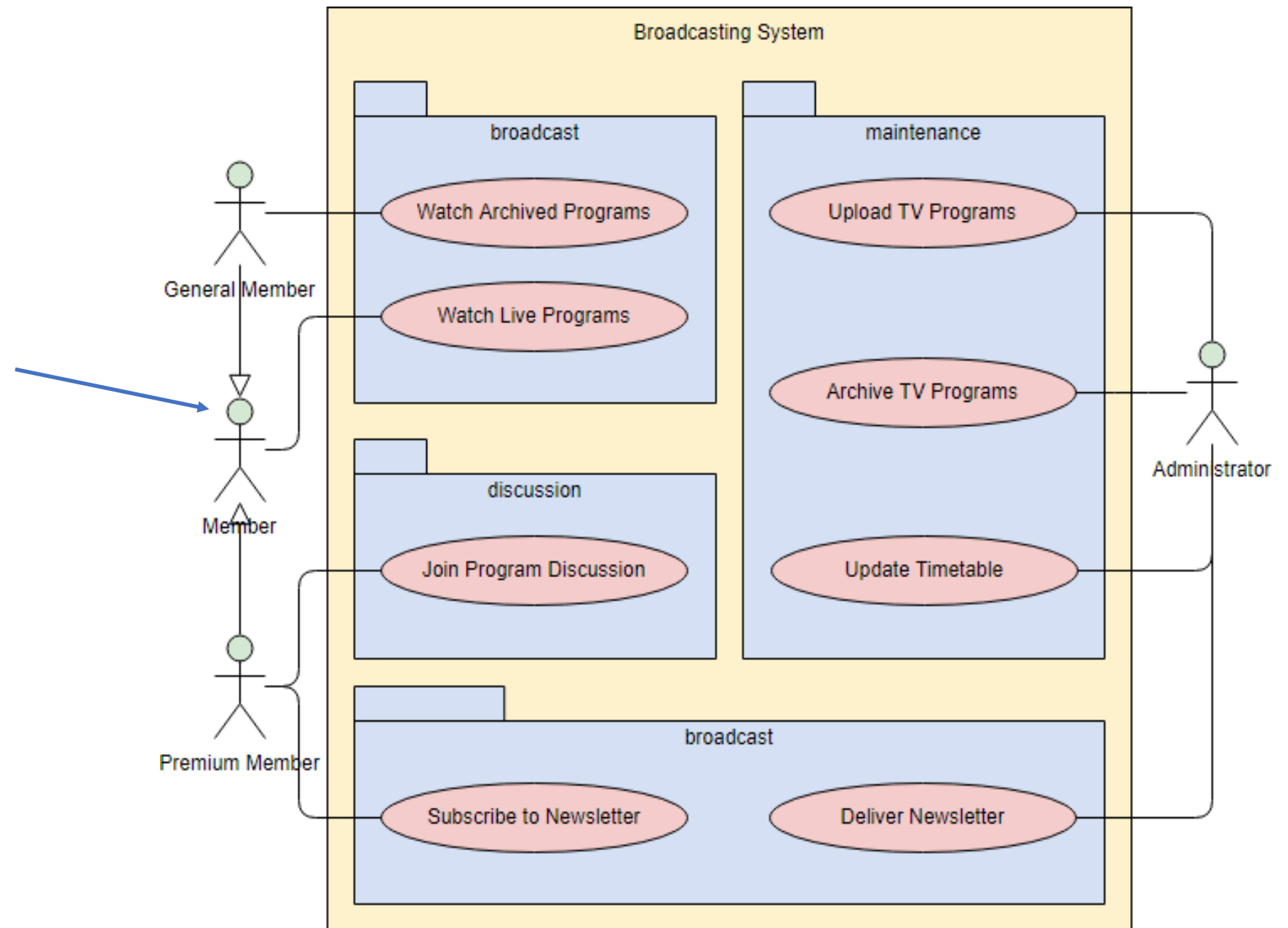
Use case diyagramları



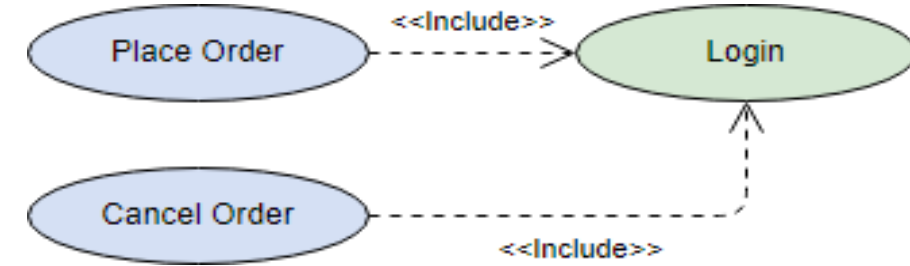
- ❑ use case diyagramı, kullanıcının belirli bir amaca ulaşmak için sistemi nasıl kullandığını açıklar.
- ❑ Diyagram , ilgili use case lerden ve aktörlerden oluşur; bunları birbirine bağlanarak şu sorular cevaplanır:
 - Ne açıklanıyor? (sistem),
 - Sistemi kim kullanıyor? (aktörler)
 - Aktörler neleri gerçekleştirmek istiyor? (use cases)
- ❑ Use case ler , gereksinimleri kullanıcı bakış açısından değerlendirir ve sistemin doğru şekilde geliştirilmesini sağlar.



Aktörler arasında genelleştirme (generalization) ilişkisi var.
Member aktörü üst aktör,
GeneralMember
Ve Premium Member aktörleri alt aktörler.

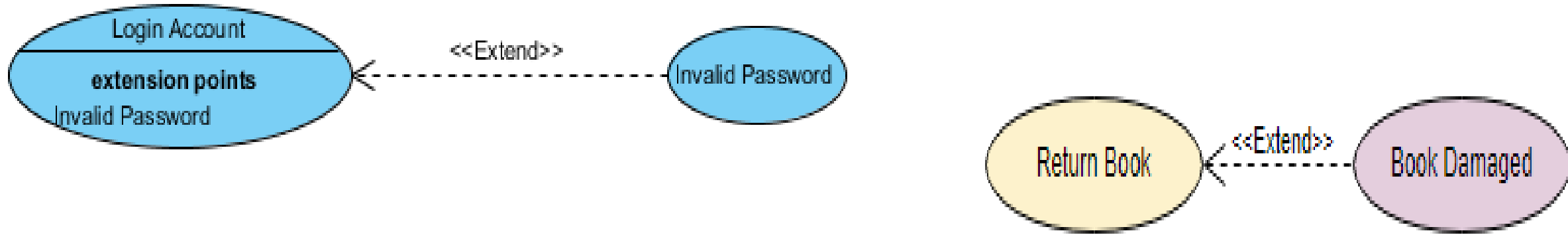


İki use case arasındaki ilişki: include ilişkisi



- ❑ Bir use case başka bir use case 'in işlevselliğini kullandığında aralarındaki ilişkiye include ilişkisi denir.
- ❑ Bir use case, iş sürecindeki akışın bir parçası olarak başka bir use case de açıklanan işlevselliği içerir.
- ❑ Temel use case tarafından alt use case tarafına bir «include " ilişkisi, temel use case bir örneğinin, alt use case senaryosunda belirtilen davranışı içereceğini gösterir.
- ❑ include ilişkisi, noktalı çizgiyle yönlendirilmiş bir okla gösterilir.

İki use case arasındaki ilişki: extends ilişkisi



❑ InvalidPassword isimli use case, temel kullanım durumu Login Account tarafından belirtilen davranışı çerebileceğini gösterir.

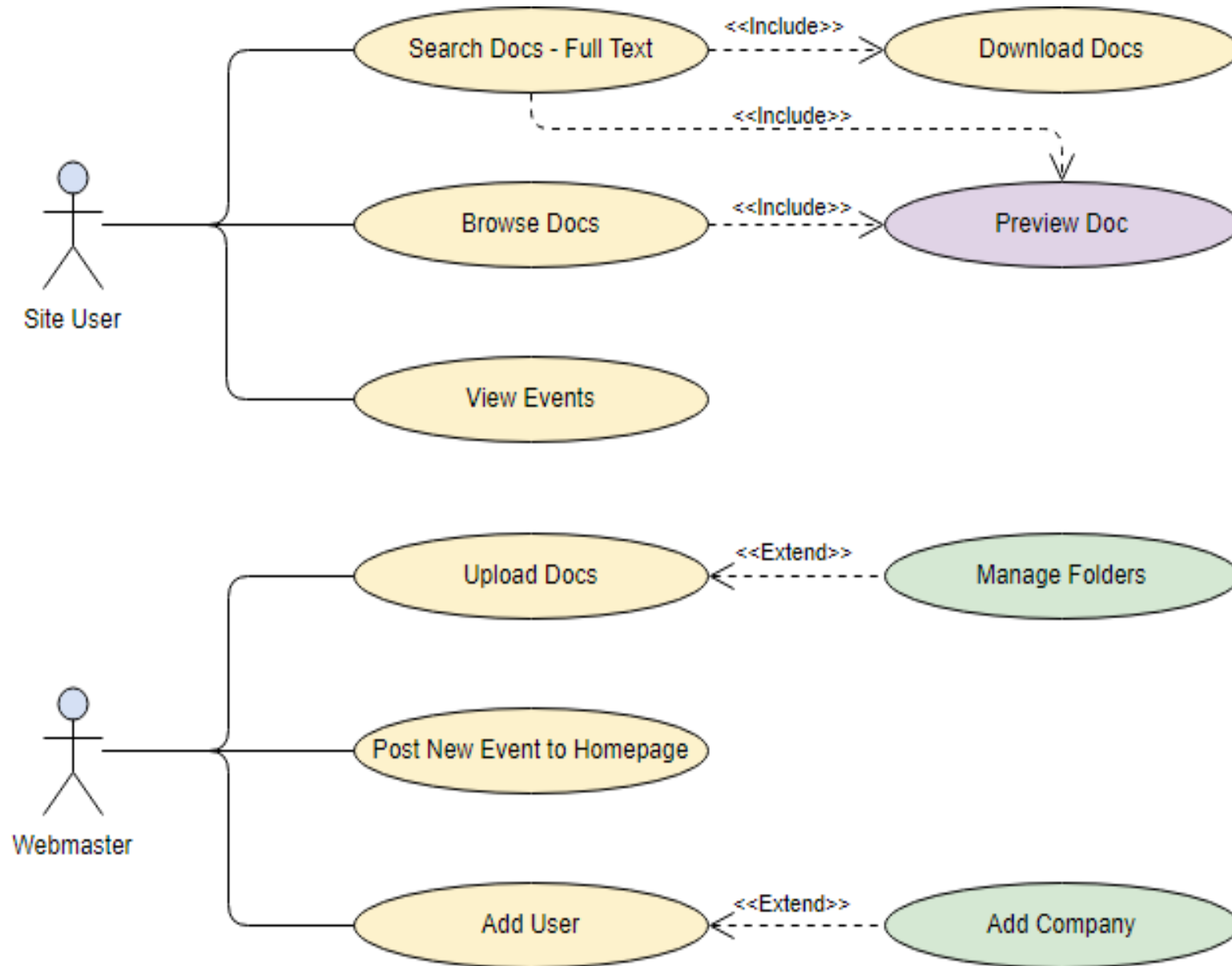
❖ Davranışın gerçekleşmesi zorunlu değildir.

❑ Noktalı çizgili ve yönlü bir okla gösterilir.

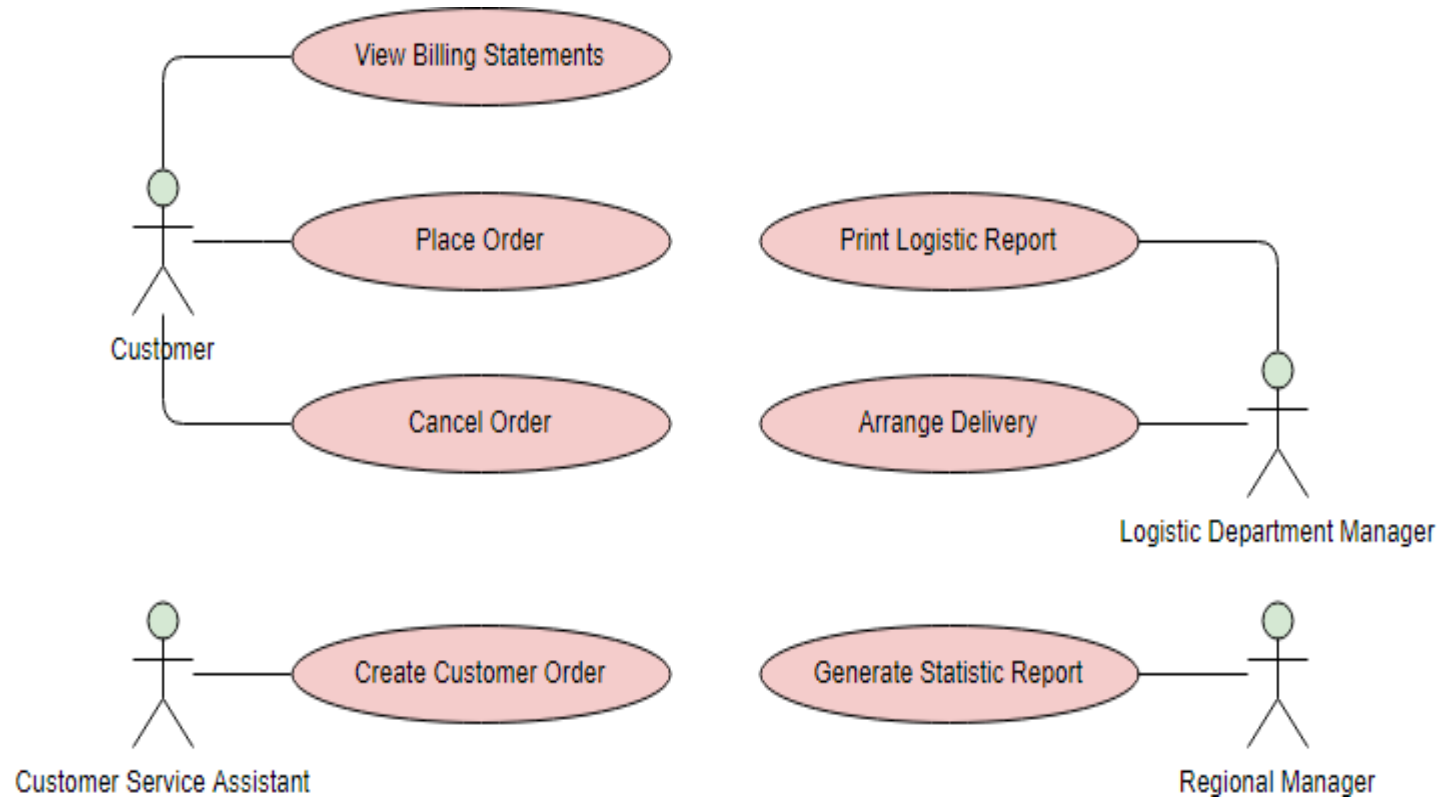
❑ Okun ucu temel use case (Login Account) gösterir ve alt use case okun tabanında (Invalid Password) bağlanır.

❑ <<extends>> kalıbı, olası bir genişletme ilişkisi olarak tanımlanır.

Document Management System (DMS) use case



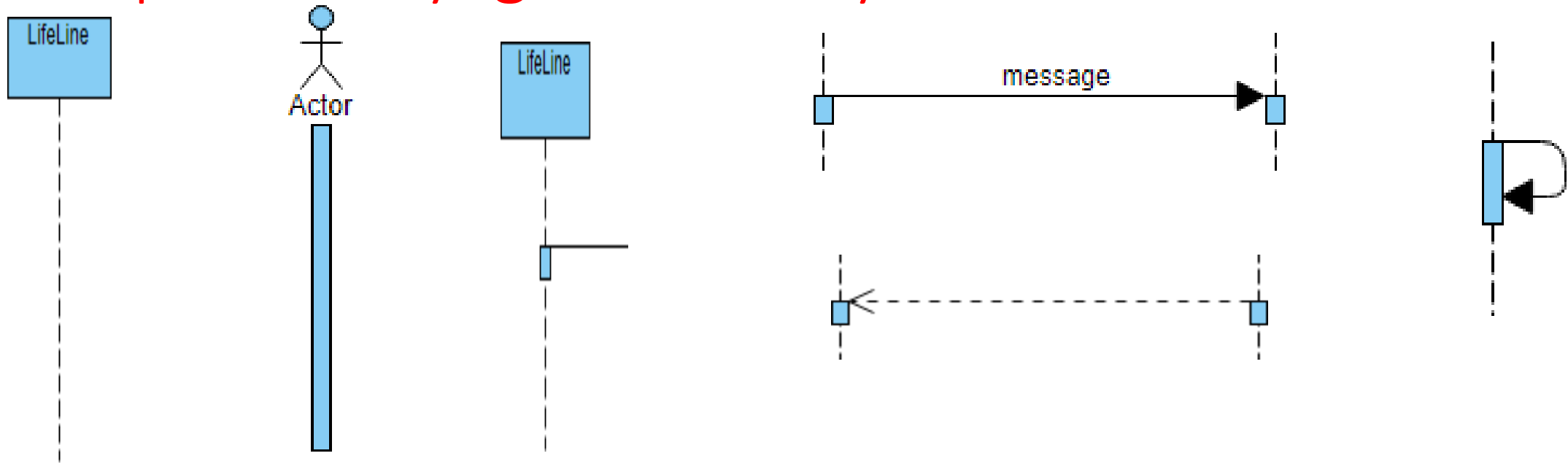
Order System use case



Sequence Diagram

- ❑ İşlemlerin nasıl gerçekleştirildiğini ayrıntılı olarak açıklayan bir etkileşim (interaction) diyagramıdır.
- ❑ Nesneler arasındaki etkileşim nesnelerin işbirliğiyle sağlanır.
- ❑ Kullanım alanlarından biri, use case olarak ifade edilen gereksinimlerden bir sonraki seviyeye geçişin sağlanmasıdır.
 - ❖ Use cases genellikle bir veya daha fazla sequence diyagramına dönüştürülür.
- ❑ Sequence diyagramları zaman odaklıdır; diyagramın dikey eksenini zamanı temsil eder; hangi mesajların ne zaman gönderildiğini gösterir.

Sequence Diagram Notasyonları

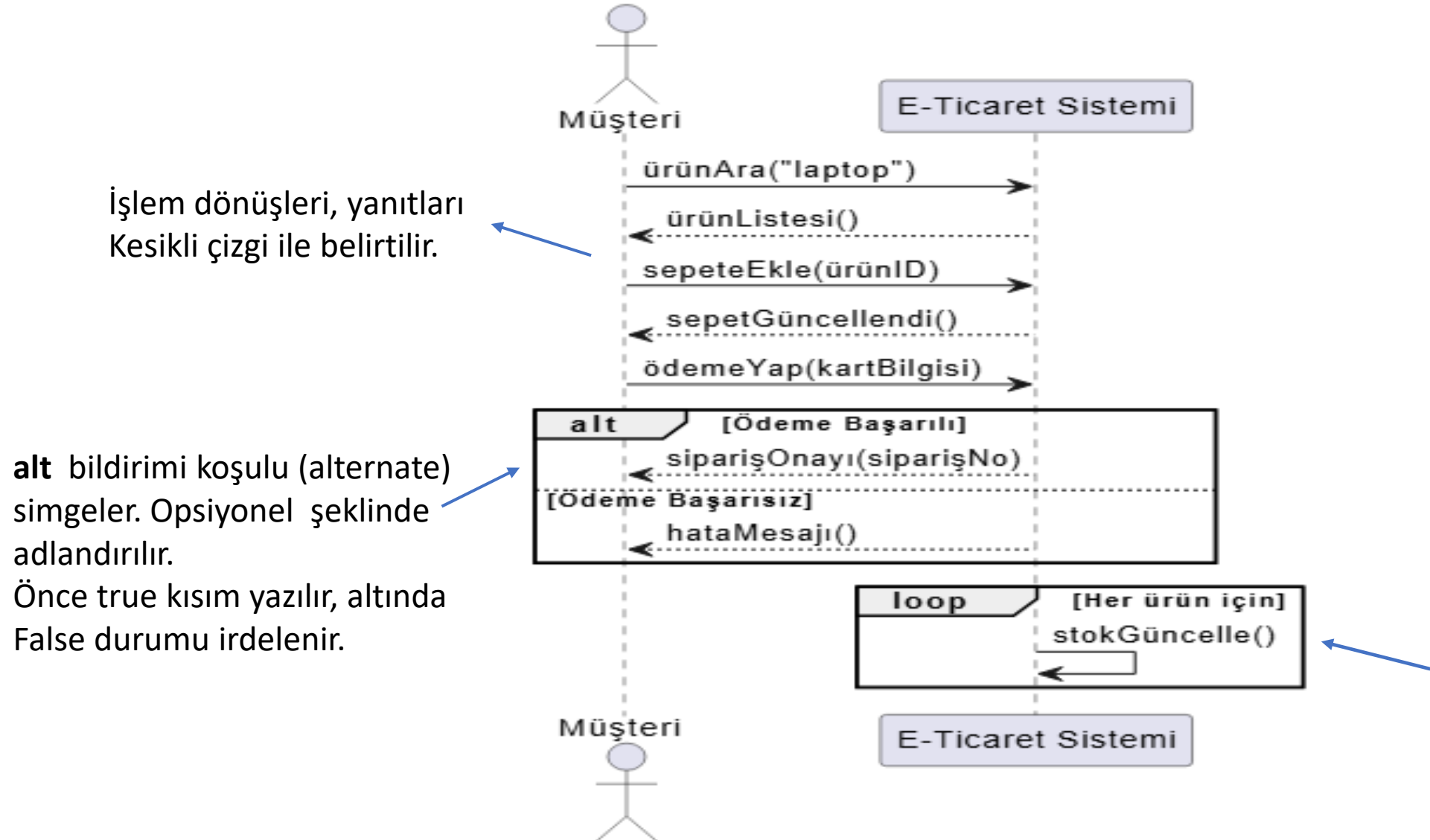


- ❑ Life Line, etkileşimdeki bireysel bir katılımcıdır.
- ❑ Aktör, özneye etkileşim kuran (örneğin, sinyal ve veri alışverişi yaparak) varlığın oynadığı roldür. İnsan kullanıcılar, harici donanımlar veya diğer özneler tarafından oynanan rollerdir.
- ❑ Aktivasyon, yaşam çizgisi üzerinde ince bir dikdörtgendir; elemanın bir işlemi gerçekleştirdiği süreyi gösterir. Dikdörtgenin üst ve alt kenarları sırasıyla başlatma ve tamamlama zamanlarını verir.

Farklı Düzeylerde Sequence Diyagramları

- ❑ Sistem kullanıcısı (aktör) ile sistem arasında, sistem ile diğer sistemler veya alt sistemler arasında gerçekleşen üst düzey etkileşimler sistem sequence diyagramlarıdır.
 - ❖ Kullanıcının (aktörün) sistemle olan etkileşimini zaman sırasına göre verilir.
 - ❖ Sistem bir kara kutu olarak alınmıştır.
 - ✓ Çünkü iç bileşenler değil, yalnızca dış olaylar ve sistem yanıtları modellenmiştir.
- ❑ Bir use case ya da işlemi diğerleri ile etkileşimli olarak gerçekleştiren etkileşim, genel sequence diyagramlarıdır.
- ❑ Nesnelerin etkileşimi temsil edildiğinde Model-Viewer-Controller MVC şablonu ifade edilir.

Sistem Sequence Diyagramı



MVC Kullanıcı Diyagramları

User (Kullanıcı): Arayüzle etkileşime giren aktör

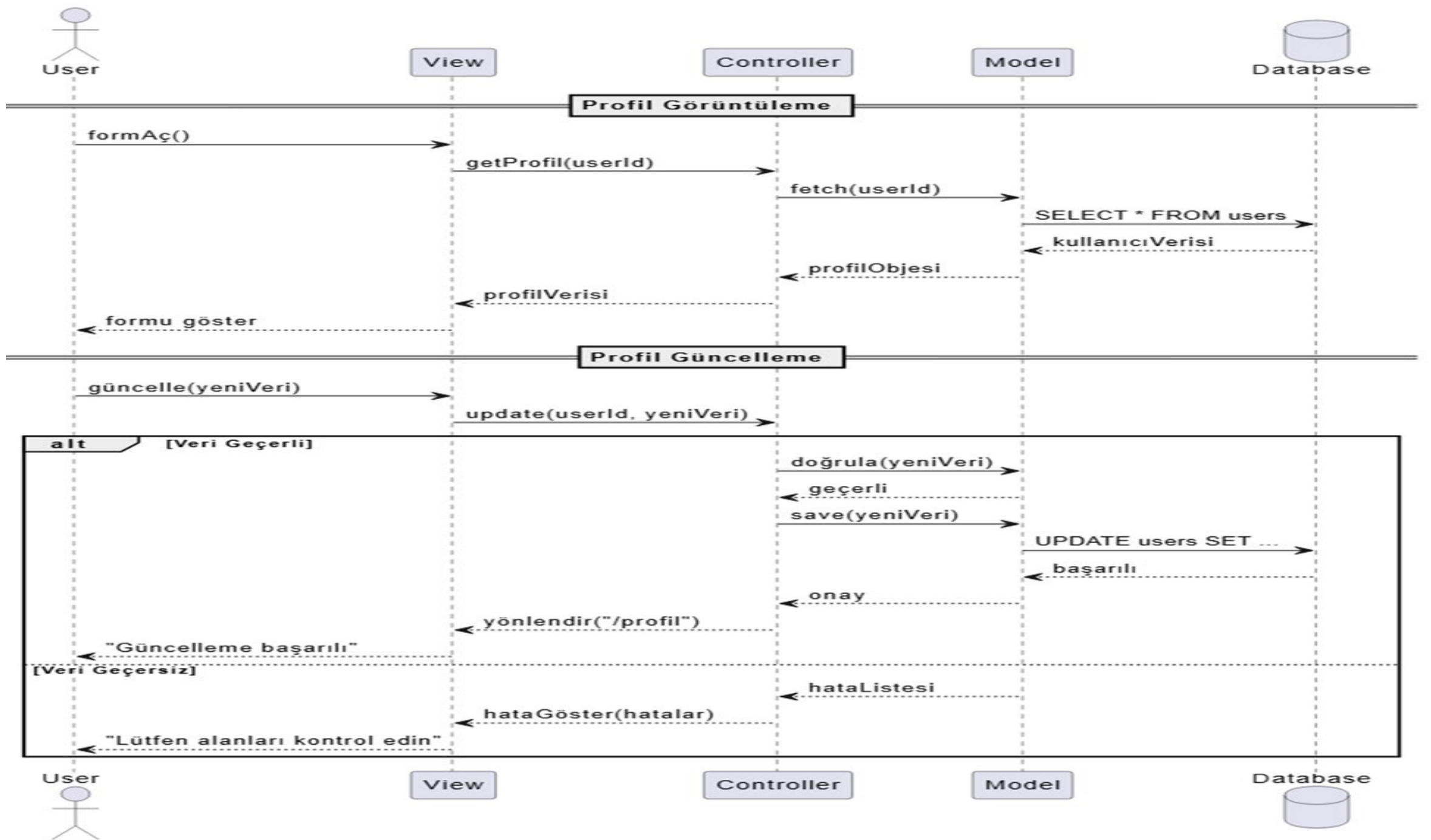
View (Görünüm): Kullanıcıya gösterilen arayüz; girdiyi alır, çıktıyı sunar

Controller (Denetleyici): Girdiyi yorumlar, Model'i günceller, View'ı seçer

Model: Veri ve iş mantığını barındırır; veritabanıyla etkileşim halindedir.

Şablonun İşleyişi:

1. View doğrudan Model ile bağlantılı olmadığından tek yönlü bağımlılık vardır. Tüm işlemler Controller üzerinden gerçekleşir. Böylece katmanlar birbirinden bağımsız test edilebilir.
2. Controller iş mantığı yazmaz; sadece doğru Model metodunu çağırır ve doğru View seçer.
alt / else bloğu, gerçek dünyadaki başarısız doğrulama senaryosunu modeller.
Alternatif akışların oluşturulması kritik sonuçlar doğurur.
3. Model katmanı doğrulama, hesaplama ve veri erişimi gerçekleştirir. Controller katmanına sızma olursa MVC bozulur



MVC Sequence Diagram

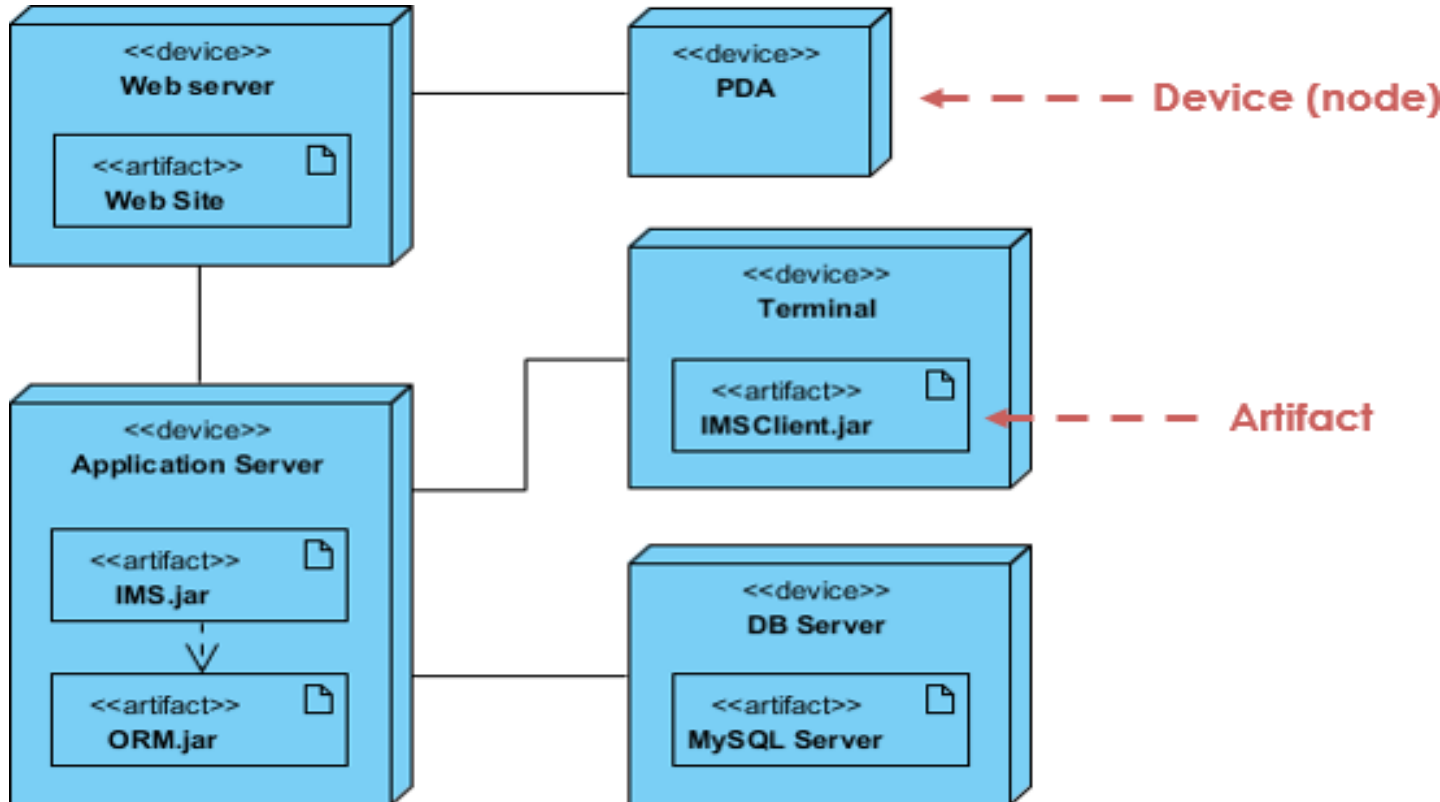
Soyut bir mimari şablonu somut bir zaman çizelgesine dönüşür.

Her istek Model –View- Controller katmanları arasında olası bir yol izler:

View → Controller → Model → Controller → View

Bileşen	Alınan Mesaj	Gönderilen Mesaj
View	Kullanıcı olayları, Controller yanıtları	Controller'a istek
Controller	View'dan istek	Model'e komut, View'a yönlendirme
Model	Controller'dan sorgu/komut	DB'ye SQL, Controller'a veri
Database	Model'den SQL	Model'e sonuç

Deployment Diagram



Nesneye Yönelik bir yazılım sisteminin fiziksel yönü modellenir.

Sistem mimarisini, yazılım bileşenlerinin dağıtımını görselleştiren yapısal bir diyagramdır.

Bileşenler, geliştirme sürecinin sonundaki somut öğeleri temsil eder.

Çalışma zamanı (run time) yapılandırması statik bir görünümde modellenir; ve uygulamadaki bileşenlerin dağıtımını görselleştirilir.

UML'de dağıtım diyagramları, bu fiziksel düğümlerin ve ilişkilerinin statik yönünü görselleştirmek ve yapım detaylarını belirtmek için kullanılır. Dağıtım diyagramları, nesne yönelimli bir sistemin fiziksel yönlerini modellemede kullanılan iki diyagram türünden biridir. Bir dağıtım diyagramı, çalışma zamanı işleme düğümlerinin ve üzerlerinde bulunan bileşenlerin yapılandırmasını gösterir. Dağıtım diyagramları, bir mimarinin statik dağıtım görünümünü ele alır. Tipik olarak bir düğümün bir veya daha fazla bileşeni içermesi bakımından bileşen diyagramlarıyla ilişkilidirler.

Component Diyagramı

- ❑ Bileşenlerin daha büyük bileşenler veya yazılım sistemleri oluşturmak üzere birbirine nasıl bağlandığını gösterir.
- ❑ Yazılım bileşenlerinin mimarileri ve aralarındaki bağımlılıklar ifade edilir.
- ❑ Bu yazılım bileşenleri arasında çalışma zamanı (run time) bileşenleri, yürütülebilir bileşenler ve kaynak kod bileşenleri bulunur.

