GEREKSİNİM MÜHENDİSLİĞİ

Gereksinim mühendisliği gereksinimleri proje boyunca ele alan, gereksinimleri yaşayan bir organizma olarak değerlendiren, yazılım mühendisliğinin bir alt dalıdır. Üzerinde çalışılmaya başlanacak olan projenin amaçlarını, boyutlarını ve etkilerini belirlemeye yönelik yapılan çalışmalar olarak ifade edilebilir.

Yazılım geliştirme süreçlerinin en önemli noktası, geliştirilecek olan sistem veya yazılım için müşterinin gereksinimlerini ortaya çıkarmak ve bunları analiz etmektir. Gereksinim mühendisliği; müşterinin bir yazılımdan beklentilerini, operasyonel ve geliştirme esnasındaki kısıtlarını da dikkate alarak ortaya koyma ve gereksinimleri belgeleme işlemlerini sistematik olarak yapabilme disiplinidir.

Yazılım geliştirme sürecinin tüm aşamalarında, kalitenin artırılması ile maliyet ve kaynak kullanımının azaltılmasına yönelik önemin artması ile birlikte bu sürecinde önemi zamanla artmıştır. Bu sürecin sistem geliştirme süreçleri içerisinde en önemli ve kritik aşamalarından biri olduğunun farkına varılmıştır. Sistem gereksinimleri tam ve doğru olarak ortaya konulmadığında, daha sonraki süreçlerde bu aşamadan dolayı ortaya çıkacak problemler maliyetleri göreceli olarak artıracak ve ortaya çıkan yazılım kaynak açısından verimli olmayacaktır.

GEREKSİNİM NEDİR?

Gereksinim kavramı, bir iş ihtiyacını karşılamak üzere paydaşların ihtiyaç duyduğu yetenekler ve özellikler olarak tanımlanabilir. Gereksinim, bir bilgi sisteminin sahip olması gereken durum ya da yetenek olarak da ifade edilebilir. Bir yazılım projesinde, müşteri ile uzlaşılmış tüm istekler ve ortaya çıkabilecek tüm kısıtlar gereksinim olarak düşünülmelidir.

Gereksinim Türleri

- Kullanıcı Gereksinimleri
- Sistem Gereksinimleri
- Özel Kısıtlara Yönelik Gereksinimler

Kullanıcı Gereksinimleri:

Doğal dil ve diyagramlarla anlatılmış, sistemin ne yapacağını ve kısıtlarını anlatan doğrudan kullanıcıya yönelik müşteriler için hazırlanmış gereksinimlerdir. Bu gereksinimler, kullanıcı senaryoları (use-cases) için temel oluştururlar. Müşterilerden elde edildikleri için teknik anlamda çok fazla detay içermezler.

Sistem Gereksinimleri:

Sistem servislerine ilişkin detaylı işlevsel tanımlamalar ile donanımsal yapılara özgü hazırlanmış gereksinimlerdir. Bu gereksinimler üst seviyede sistem mimarisini tanımlarlar. Yapısal bir doküman olan sistem gereksinimleri, müşteri ile yapılan anlaşma ve sözleşmeye göre hazırlanır.

Özel Kısıtlara Yönelik Gereksinimleri:

Yazılımın performansı ve niteliklerini tanımlayan sübjektif olmayan kesin sınırlar ve kısıtlardır. Yazılımın çalışma sahası yani etki alanını belirleyen gereksinimlerdir. Yazılımın geliştirilmesine yönelik kısıtlarda bu tip gereksinimlerdir.

ETKİ ALANLARINA VE NİTELİKLERİNE GÖRE GEREKSİNİMLER;

- İşlevsel (Functional) Gereksinimler
- İşlevsel Olmayan (Non-Functional) Gereksinimler

İşlevsel (Functional) Gereksinimler

Geliştirilecek yazılımın hangi işlemleri yapacağını, hangi girdileri alacağı ve hangi çıktıları üreteceğine ilişkin gereksinimlerdir. Bu gereksinimler, sistemin sunacağı hizmetler ile işlevsel altyapısını tanımlarlar. Geliştirmeden bağımsız çoğunlukla giriş, çıkış arabirimleri, süreçler ile hata yönetimine yönelik gereksinimlerdir. İşlevsel gereksinimler, sistemin neler yapacağını soyut olarak değil de detaylandırılmış biçimde tanımlanmasını sağlarlar. **Özetle, 'geliştirilecek sistem ne yapacak?' sorusunun cevabını vermeye çalışır.**

Geliştirilecek bir hastane otomasyon sistemi ile ilgili işlevsel gereksinimlere örnek olarak aşağıdakiler verilebilir:

- Geliştirilecek yazılım, bir hastaya ait bütün kimlik verilerini tutacaktır.
- Geliştirilecek yazılım üzerinden hasta uygun tarih ve saate göre randevu alacaktır.
- Geliştirilecek yazılım, hastaya otomatik olarak uygun bir randevu numarası verecektir.
- Geliştirilecek yazılım üzerinden doktor hastalara ilişkin tanı bilgilerini sisteme girebilecektir.
- Geliştirilecek yazılım üzerinden doktor, hastaya reçete yazabilecektir.
- Doktor geliştirilecek yazılım üzerinden hastanın tahlil ve laboratuvar sonuçlarını görüntüleyebilecektir.
- Geliştirilecek yazılım, hastanedeki poliklinik bilgilerini tutacaktır.

İşlevsel Olmayan (Non-Functional) Gereksinimler

Geliştirilecek yazılımın daha çok kısıtları ile ilgili fiziksel ortam, arayüzler, kullanıcı odaklı olma, yanıtlama zamanı, performans, güvenlik, güvenilirlik ve kalite güvence gibi soyut niteliklerini belirleyen gereksinimlerdir.

Geliştirilecek bir hastane otomasyon sistemi ile ilgili işlevsel olmayan gereksinimlere aşağıdakiler örnek verilebilir:

- Geliştirilecek yazılım, web ortamında çalışacaktır.
- Geliştirilecek yazılım, tüm web tarayıcılar tarafından desteklenecektir.

- Geliştirilecek sistemi 1000 kişi aynı anda kullanırken tepki süresi 3 saniyeyi geçmeyecektir.
- Geliştirilecek sistem yetkisiz kullanıcı girişlerine yönelik %100 güvenli olacaktır.
- Geliştirilecek sistem üzerinde herhangi bir sorguya yönelik yanıtlama zamanı 2 saniyeden daha kısa olacaktır.

GEREKSİNİM MÜHENDİSLİĞİ SÜREÇLERİ NELERDİR?

Gereksinim mühendisliğinin dört temel süreci ve bunları gerçekleştirmek için tüm süreçlere uygulanan faaliyetler vardır. Bu süreçler birbiri ile bütenleşiktir. Süreçlerin tamamının yazılım geliştirme sürecinde uygulanması gerekmez. Organizasyon ve proje büyüklüğüne göre uygun olanlar seçilir. Bunlar;

- Fizibilite (yapılabilirlik) analizi,
- Gereksinim analizi,
- Gereksinim tanımlama,
- Ayrıntılı gereksinim belirtimidir.

FİZİBİLİTE (YAPILABİLİRLİK) ANALİZİ

Fizibilite çalışması, müşteri gereksinimlerine ait maliyetin, uygun bütçe ve teknoloji sınırları içinde olup olmadığının tespit edildiği süreçtir. Müşterilere, fazladan istenilen hizmetlerin bir ek maliyeti olduğu ve bunun hem zaman hem de maddi açıdan yeterli olması gerektiği belirtilmelidir. Ayrıca, kar-maliyet analizinin de çıkarılması ve belli kriterlere karar verilmesi gereklidir. **Fizibilite çalışması**, bir yazılım ya da sistem geliştirme projesinin başında yapılan, projenin **gerçekleştirilebilirliğini** değerlendiren önemli bir analiz sürecidir. Gereksinim mühendisliği sürecinin ilk aşamalarında yapılan fizibilite çalışması, projenin **teknolojik, finansal, zaman ve kaynak açısından uygulanabilirliğini** belirler.

Teknik Fizibilite

Projede kullanılacak teknoloji ve altyapı gereksinimlerinin mevcut sistemle uyumlu olup olmadığının değerlendirilmesidir.

Örnek: Yeni bir yazılım geliştirme projesinde, kullanılacak programlama dillerinin ve yazılım çerçevelerinin mevcut sistemler ve ekip becerileriyle uyumlu olup olmadığının değerlendirilmesi.

Finansal Fizibilite

Projenin finansal açıdan yapılabilirliğini değerlendirir. Projenin toplam maliyeti, kaynak gereksinimleri, iş gücü, altyapı, yazılım lisansları gibi kalemler dikkate alınır.

Örnek: Yeni bir yazılım geliştirme projesi için gerekli olan yazılım lisansları, donanım yatırımları ve iş gücü maliyetlerinin toplamı, mevcut bütçe ile karşılaştırılır.

Operasyonel Fizibilite

Projenin mevcut iş süreçleri ve organizasyon yapısı ile uyumlu olup olmadığını değerlendirir. Ayrıca, projenin başarısının işletme içindeki diğer ekipler tarafından kabul edilip edilmeyeceğini araştırır.

Örnek: Yeni bir CRM yazılımının, satış ekibi tarafından nasıl kullanılacağını, veri girişi, eğitim ihtiyaçlarını ve destek süreçlerini değerlendirmek.

Hukuki ve Yasal Fizibilite

Projenin yerel, ulusal ve uluslararası yasalarla uyumlu olup olmadığını araştırır. Bu, veri gizliliği yasaları, telif hakları, patentler ve düzenleyici gereklilikler gibi konuları içerir.

Örnek: Bir sağlık yazılımı geliştiriliyorsa, sağlık verileriyle ilgili yasal düzenlemelere (örneğin, HIPAA, GDPR) uygunluk kontrol edilir.

Zaman Fizibilitesi:

Projenin belirlenen sürede tamamlanıp tamamlanamayacağı değerlendirilir. Proje süresinin ne kadar uzun olacağı ve belirli tarihlerde tamamlanması gereken kilometre taşlarının ne kadar gerçekçi olduğu belirlenir.

Örnek: Yeni bir e-ticaret platformu geliştirilmesi projesinin, yıl sonuna kadar piyasaya sürülmesi için gereken süre, ekip büyüklüğü ve kaynaklar göz önüne alınarak hesaplanır.

Fizibilite çalışması, bir yazılım ya da sistem geliştirme projesinin başarısının temelini atar. Gereksinim mühendisliğinin bir parçası olarak yapılan bu çalışma, projeye başlamadan önce projenin potansiyelini ve risklerini anlamanızı sağlar. Bu, sadece zaman ve maliyetin doğru yönetilmesini sağlamakla kalmaz, aynı zamanda projedeki belirsizlikleri en aza indirir ve tüm paydaşlar için daha sağlıklı bir karar alma süreci oluşturur.

BAŞABAŞ NOKTASI (KARA GEÇİŞ) ANALİZİ

Bu analiz herhangi bir yatırım kararında kullanılmaktadır. Girişimciler işyeri kurarlarken veya pazarlamacılar yeni ürün piyasaya sürerken nasıl ne kadar zamanda kar etmeye başlarım sorusunu soruyorsa, bilişimcilerde yatırım kararlarında (yazılım geliştirme vb.) ne zaman başabaşı bulurum sorusunu sorarlar. İşte bu analiz ile toplam gelirin toplam gidere eşit olduğu (başabaş) noktasının bulunması hedeflenir. Böylece işletme, kaç birim üretimde bulunduğunda masraflarını karşılayabileceğini belirlemeye çalışır. Üretim miktarı ve birim fiyatlarda oynayarak istenilen senaryo yaratılabilir. Yapılacak duyarlılık analizi ile oluşturulan değişik senaryolar başka kriterler de dikkate alınarak doğru kararı vermemizde yardımcı olur.

Gereksinim mühendisliği sürecinin bir parçası olarak yapılan fizibilite çalışmasında başabaş analizi önemli bir araçtır, çünkü projelerin sadece teknik açıdan değil, finansal açıdan da değerlendirilmeleri gerekir. Bu analiz, projenin kârlılık noktalarını, maliyetleri ve gelirleri analiz ederek projenin sürdürülebilirliğine dair önemli bilgiler sağlar.

$$\text{Başabaş Noktası (Satış Miktarı)} = \frac{\text{Toplam Sabit Maliyetler}}{\text{Birim Başına Satış Fiyatı} - \text{Birim Başına Değişken Maliyet}}$$

SENARYO: YENİ BİR YAZILIM ÜRÜNÜ GELİŞTİRME

Bir yazılım geliştirme şirketi, yeni bir **muhasebe yazılımı** geliştirmeyi planlıyor. Yazılım, küçük ve orta ölçekli işletmelerin finansal süreçlerini yönetmelerine yardımcı olacak. Şirket, bu yazılımı bir **abonelik modeliyle** satmayı planlıyor. Şirketin hedefi, bu yazılımı geliştirip piyasaya sunduktan sonra sürdürülebilir bir gelir akışı elde etmektir.

Projenin Temel Parametreleri:

Yazılımın Satış Fiyatı: Aylık 50 TL (abone başına)

Sabit Maliyetler (Geliştirme, Pazarlama, vb.): 500.000 TL (ilk yıl için)

Değişken Maliyetler (Birim başına maliyet): 10 TL (her bir abone için)

İlk Yıl Beklenen Satış Hedefi: 10.000 abone

Yazılımın Geliştirilmesi ve Pazarlanması İçin Yapılacak Yatırım: 500.000 TL

Gelir Tahminleri: Şirket, yazılımın her bir abone için **50 TL/ay** ücret almayı planlıyor. Aylık abonelik gelirini yıllık olarak hesaplayalım:

$$\text{Yıllık Gelir} = 50\,\text{TL/abone/ay} \times 10.000\,\text{abone} \times 12\,\text{ay}$$

$$\text{Yıllık Gelir} = 50 \times 10.000 \times 12 = 6.000.000\,\text{TL}$$

Yani, yazılımın ilk yıl sonunda toplam **6.000.000 TL** gelir elde edilmesi bekleniyor.

Maliyet Hesaplamaları:

Sabit Maliyetler (Geliştirme, Pazarlama vb.): 500.000 TL (projenin başlangıcında yapılan yatırımlar)

Değişken Maliyetler (Birim başına maliyet): 10 TL (her bir abone için yıllık maliyet)

Toplam Değişken Maliyetler:

$$\text{Değişken Maliyetler} = 10\,\text{TL/abone} \times 10.000\,\text{abone} = 100.000\,\text{TL}$$

Toplam Maliyetler (Sabit + Değişken):

$$\text{Toplam Maliyetler} = 500.000\,\text{TL} + 100.000\,\text{TL} = 600.000\,\text{TL}$$

Başabaş Noktasının Hesaplanması:

Başabaş noktasını bulmak için gelirlerin ve maliyetlerin eşitlendiği noktayı bulmamız gerekiyor. Başabaş noktası, aşağıdaki şekilde hesaplanır:

$$\text{Başabaş Noktası} = \frac{\text{Toplam Sabit Maliyetler}}{\text{Birim Başına Satış Fiyatı} - \text{Birim Başına Değişken Maliyet}}$$

Burada:

Sabit maliyetler: 500.000 TL

Birim başına satış fiyatı: 50 TL

Birim başına değişken maliyet: 10 TL

$$\text{Başabaş Noktası} = \frac{500.000}{50 - 10} = \frac{500.000}{40} = 12.500 \text{ abone}$$

Başabaş Noktası: 12.500 abone

Yani, yazılım şirketinin başabaş noktasına ulaşabilmesi için **12.500 abone**ye ulaşması gerekiyor. Bu, şirketin sabit maliyetleri (geliştirme, pazarlama) ve değişken maliyetleri karşılayabilmesi için gereken minimum abone sayısıdır.

Başabaş Noktasının Yorumlanması:

- Şirket, başabaş noktasına ulaştığında, sadece giderlerini karşılamakla kalmayacak, herhangi bir kâr elde etmeye başlayacak.
- Şirketin ilk yıl için hedeflediği 10.000 abone sayısı, başabaş noktasının altında kalıyor. Yani, bu hedefle şirket yalnızca giderlerini karşılayacak ve kâr elde etmeye başlamayacak.
- Eğer şirket, abone sayısını 12.500'e çıkarabilirse, gelirler kar üretmeye başlayacak.

GEREKSİNİM ANALİZİ

Müşterinin ve tüm paydaşların sistemden olan beklentilerini ortaya çıkarmak ve bunları uygun metodojilerle ayrıştırıp arındırma sürecidir. Müşterinin gereksinimlerini doğru almak ve analiz etmek, projenin tamamlanmasında en önemli kriterdir. Gereksinimlerin önceliklendirilmesi bu süreçte yapılır. Gereksinim mühendisliği süreçlerinde bazı analiz araçları ve yöntemler, proje gereksinimlerinin doğru bir şekilde toplanması, analiz edilmesi ve yönetilmesi için kullanılabilir. Balık kılçığı diyagramı, SWOT analizi ve Gantt şeması gibi araçlar doğrudan gereksinim analizinin ana unsurları olmasa da, belirli durumlarda bu araçlar yardımcı olabilir.

Balık Kılçığı Diyagramı (Ishikawa Diyagramı / Sebep-Sonuç Diyagramı)

Bir problemi analiz etmek ve o probleme yol açan olası **kök nedenleri** belirlemek için kullanılan bir araçtır. Adını, diyagramın şekli balık kılçığına benzer olduğu için alır; bu diyagramda bir **baş** (problem veya sonuç) ve ona bağlı **kemikler** (nedenler) vardır. Bu diyagram, bir problem veya hedefe ulaşmak için öncelikle tüm olası sebepleri sistematik bir şekilde incelemeyi amaçlar.

Balık Kılçığı Diyagramının Amaçları

Kök Neden Analizi: Bir problem yaşandığında, yüzeysel nedenlerden çok derindeki sebepleri anlamak önemlidir. Problemin altında yatan esas nedenleri daha kolay bir şekilde keşfetmeye yardımcı olur.

Sorun Çözme: Belirli bir sorunun çözümüne yönelik adımlar atılmadan önce, olası tüm nedenlerin araştırılması gerekir. Bu diyagram, sorunun çözülmesinde hangi alanlara odaklanılması gerektiğini gösterir.

Ekip Çalışması ve Beyin Fırtınası: Ekip üyeleri, problemle ilgili olası tüm nedenleri listeleyebilir ve daha sonra bu nedenlerin çözülmesi için çeşitli çözüm önerileri geliştirilebilir.

Süreç İyileştirme: Üretim, yazılım geliştirme veya hizmet süreçlerinde kalite kontrolü ve sürekli iyileştirme için balık kılçığı diyagramı kullanılarak süreçlerdeki zayıf noktalar tespit edilebilir.

ÖRNEK SENARYO: YAZILIMDA PERFORMANS SORUNU

Bir e-ticaret platformunun yazılımında, **yavaş sayfa yükleme süresi** ve **düşük performans** şikayetleri alınıyor. Proje yöneticisi bu problemi çözmeye karar veriyor ve bunun için bir **balık kılçığı diyagramı** oluşturuyor.

Problem (Baş):

- Yavaş sayfa yükleme süresi ve düşük performans.

Ana Kategoriler (Kemikler):

- İnsan
- Makine
- Malzeme
- Metod
- Ölçümler

YAZILIMDA PERFORMANS SORUNU

Bir e-ticaret platformunun yazılımında, yavaş sayfa yükleme süresi ve düşük performans şikayetleri alınıyor.

İnsan Kategorisi

- Eksik Eğitim: Geliştiricilerin performans optimizasyonu konusunda yeterli eğitim almadığı için, yazılımda performans iyileştirmeleri yapılmıyor olabilir.
- Deneyimsiz Geliştiriciler: Deneyimsiz ekip üyelerinin kodda performans sorunlarına yol açabilecek hatalar yapması.
- İletişim Sorunları: Takım içi etkili iletişim eksiklikleri, performans iyileştirmeleri için gerekli değişikliklerin yapılmaması.

Makine Kategorisi

- Donanım Yetersizliği: Sunucuların donanım kapasitesinin yazılımın ihtiyaçlarını karşılamaması.
- Sunucu Performansı: Web sunucularının yeterince hızlı olmaması, düşük bant genişliği veya ağ gecikmeleri.

Malzeme Kategorisi

- Yazılımın Eski Sürümü: Eski sürümlerde performans iyileştirmeleri yapılmamış olabilir. Yeni sürümlerin daha iyi optimizasyon sunması bekleniyor.
- Framework Sorunları: Kullanılan framework'ün performans sorunlarına yol açması (örneğin, gereksiz kaynak tüketimi).

Ölçümler Kategorisi

- Performans ölçümlerinin yapılmaması: Performans ölçümlerinin yapılmaması veya doğru bir şekilde kaydedilmemesi, sorunun nereden kaynaklandığını belirlemeyi zorlaştırır.
- Yetersiz Logs ve Hata Raporları: Yetersiz hata raporlaması veya log dosyalarının eksik olması, sorunların tespiti ve çözülmesini engeller.

SWOT ANALİZİ

SWOT analizi, projenin güçlü ve zayıf yönlerini belirlemek ve dış çevreden kaynaklanan tehdit ve fırsatları tespit edip bunlara karşı önlem almak için kullanılan bir yöntemdir.

- **S (Strenghts):** İşletmenin güçlü/üstün olduğu yönlerin tespit edilmesi demektir.
- **W (Weakness):** İşletmenin güçsüz/zayıf olduğu yönlerin tespit edilmesidir.
- **O (Opportunities):** İşletmenin sahip olduğu fırsatları ifade eder.
- **T (Threats):** İşletmenin karşı karşıya olduğu tehdit ve tehlikelerdir.

SWOT analizi, hedeflerin ve projeyle ilgili tüm faktörlerin uygun şekilde tanımlanmasını sağlayan bir iş analizi sürecidir. Bu analiz yapılırken hem dahili hem de harici bileşenler dikkate alınır, çünkü her ikisi de bir projenin veya girişimin başarısını etkileme potansiyeline sahiptir.

Örnek Senaryo: Bir yazılım geliştirme şirketi, küçük ve orta ölçekli işletmelerin (KOBİ) envanterlerini dijital ortamda takip edebilmelerini sağlayacak bulut tabanlı bir stok yönetim sistemi geliştirmeyi planlıyor.

Bu platform, kullanıcıların stoklarını daha verimli bir şekilde yönetmelerine, sipariş süreçlerini hızlandırmalarına ve envanter hatalarını azaltmalarına yardımcı olacak. Projenin **SWOT analizi**ni gerçekleştirelim.

Güçlü Yönler (Strengths)

Deneyimli Geliştirici Ekibi: Yazılım ekibi, bulut teknolojileri ve veritabanı yönetimi konularında deneyime sahip. Ayrıca, yazılım mimarisi konusunda güçlü bir altyapıya sahip.

Esnek Altyapı ve Ölçeklenebilirlik: Sistem, kullanıcı sayısının artması durumunda kolayca ölçeklendirilebilecek şekilde tasarlanacak. Kullanıcıların işletme ihtiyaçlarına göre modüler özellikler eklenebilir.

Kullanıcı Dostu Arayüz: Platform, kullanıcı deneyimini ön planda tutarak, stok yönetimi ve sipariş takibi gibi işlemleri basitleştiren ve hızlı erişim sağlayan bir tasarıma sahip olacak.

Raporlama ve Analiz: Geliştirilecek sistem, kullanıcıların stok durumu, satış trendleri, ürün talepleri gibi verileri analiz edebilmesine olanak tanıyacak.

Zayıf Yönler (Weaknesses)

Sınırlı Pazar Araştırması: Şirket, yeni bir pazarın içine girmeyi planladığı için potansiyel kullanıcı kitlesine dair pazar araştırması yeterince derinlemesine yapılmamış olabilir. Bu da platformun beklentilere tam olarak hitap etmeyebilir.

Küçük Ekip ve Kaynak Kısıtlamaları: Şirketin yazılım geliştirme ekibi, sınırlı sayıda geliştiriciden oluşuyor, bu da geliştirme sürecini uzatabilir ve projede bazı aksaklıklar yaşanmasına neden olabilir.

Entegrasyon Zorlukları: Yeni sistemin eski yazılımlar veya mevcut veritabanlarıyla entegrasyonu zor olabilir. Bu da müşterilerin sisteme geçiş sürecinde zorluk yaşamasına neden olabilir.

Zaman Kısıtlamaları: Proje, belirli bir zaman diliminde tamamlanması gereken bir hedefe sahip olabilir. Bu, özellikle kaliteli test süreçlerinin atlanması veya özelliklerin aceleye getirilmesi gibi riskler doğurabilir.

Fırsatlar (Opportunities)

Bulut Tabanlı Çözümlere Yönelik Artan Talep: KOBİ'lerin dijital dönüşümü hızla benimsemesi ve bulut tabanlı yazılımlara geçiş yapma eğilimleri, platforma olan talebi artırabilir.

Veri Analitiği ve Yapay Zeka (AI) Kullanımı: Stok yönetiminde büyük veri ve yapay zeka çözümleri kullanarak, ürün taleplerini tahmin etme, envanter optimizasyonu ve talep analizi gibi özellikler eklemek platformun değerini artırabilir.

Mobil Uyumluluk ve Uygulama Geliştirme: Mobil cihazların yaygınlaşmasıyla birlikte, mobil uyumlu bir uygulama geliştirerek kullanıcıların stoklarına her yerden erişebilmesi sağlanabilir.

Büyük Veri ve IoT Entegrasyonu: Internet of Things (IoT) teknolojisi ile bağlantılı cihazlardan gelen verilerin entegrasyonu, stok yönetimini daha da verimli hale getirebilir. Örneğin, sensörler aracılığıyla gerçek zamanlı stok takibi yapılabilir.

Tehditler (Threats)

Yoğun Rekabet: Bulut tabanlı yazılım çözümleri pazarında birçok rakip bulunuyor. Özellikle büyük firmalar, çok daha güçlü altyapılar ve geniş müşteri portföyleriyle pazara hakim olabilir.

Teknolojik Değişiklikler ve Yenilikler: Teknolojinin hızla değişmesi ve yeni yazılım çözümlerinin ortaya çıkması, platformun kısa sürede eskimesine veya kullanıcı taleplerine yeterince cevap verememesine neden olabilir.

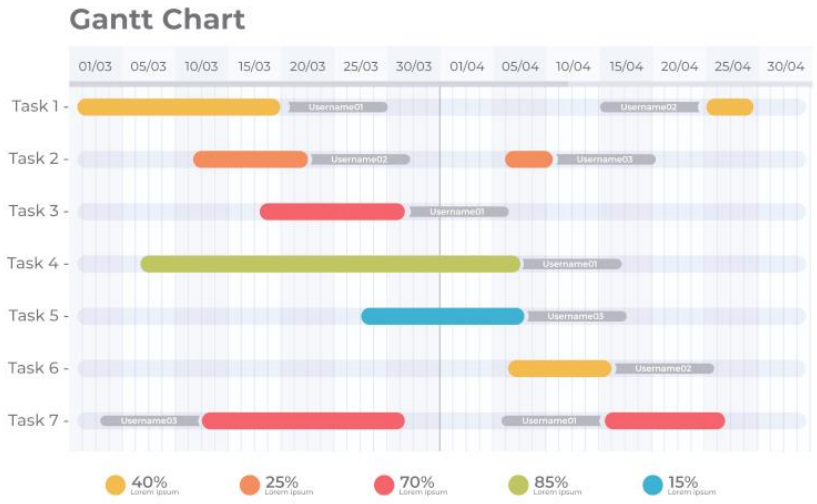
Veri Güvenliği ve Gizlilik: Bulut tabanlı sistemlerde veri güvenliği büyük bir endişe kaynağıdır. Kullanıcıların kişisel ve finansal bilgileri, özellikle kötü niyetli saldırılara karşı korunmalıdır.

Yasal Düzenlemeler ve Uyumluluk: Yasal düzenlemeler, özellikle veri koruma ve gizlilikle ilgili kurallar yazılımın işleyişini etkileyebilir. Bu düzenlemelere uyum sağlamak, platformun gelişimini zorlaştırabilir.

GANTT ŞEMASI

Bir projenin zaman çizelgesini görsel olarak gösteren, proje yönetiminde yaygın olarak kullanılan bir araçtır. Bu şema, projenin başlangıç ve bitiş tarihlerini, görevlerin ve alt görevlerin sürelerini, birbirleriyle olan ilişkilerini ve projedeki ilerlemeyi net bir şekilde görmeyi sağlar.

Gantt şeması, projelerin planlanması, zaman yönetimi, ilerleme takibi ve kaynakların etkin kullanımı açısından önemli bir araçtır. Özellikle karmaşık projelerde, proje yöneticilerine ve ekip üyelerine rehberlik sağlar. Bu sayede projelerin başarıyla tamamlanma olasılığı artırılabilir.



GEREKSİNİM TANIMLAMA

Gereksinim tanımlama süreci, öncelikli olarak geliştirilmesine karar verilen gereksinimlerin müşterinin anlayabileceği formlara dönüştürerek gerekli tanımlamaların yapıldığı süreçtir. Bu süreçte UML (Unified Modeling Language – Birleşik Modelleme Dili) diyagramları veya formlar kullanılır. Görsel anlamda yapısal bir doküman hazırlanır. Gereksinim tanımlama, müşteriye yönelik bir süreçtir ve gereksinim analizi süreciyle iç içe yürütülür. Gereksinimleri tanımlama aşamasında, müşteri ile yapılan pazarlık sonucu üzerinde uzlaşılan gereksinimler kağıda dökülür. Gereksinimleri tanımlarken; konuşma dili ile yazılmış belgeler, use-case'ler(kullanıcı senaryoları), kullanım şemaları, formel modeller (matematiksel gösterim) ve bir ön ürünün tasarlanması gibi birçok tanımlama aracı kullanılır. Birden fazla tanımlama aracı birlikte de kullanılabilir.

AYRINTILI GEREKSİNİM BELİRTİMİ

Müşteri gereksinimlerine ilişkin detayları tanımlama, ortak ve ortak olmayan noktaları ortaya koyma sürecidir. Yazılım tasarımına giriş yapılır. Bu süreç daha çok uygulama geliştiriciler ile müşteri ortamındaki teknik paydaşlar içindir. Gereksinim tanımlama sürecinden ortaya çıkan formların her biri için detaylı formlar ve UML diyagramları oluşturulur.