Beykent Üniversitesi
Yazılım Mühendisliği Bölümü
Yazılım Mühendisliği Tasarım Projesi
Grup1

Rapor 1
2024 – GÜZ

Proje İsmi (kısaltması ile birlikte)

Grup Elemanları
Öğrenci No, Adı Soyadı (Alfabetik Sırada)

I Projenin Tanımı (16 font)

1.Projenin Amacı ve Hedefleri (14 font)

Amaçlar, projede çözümlenmesi hedeflenen problemi en üst düzeyde tanımlar. Bu nedenle projenin tek bir amacı olacaktır. Hedef ise projede sonunda gerçekleştirileceklere yönelik sıralanışlardır. Amaçlardan daha kesin olan hedefler projenin tamamlanması ile değerlendirilecek niceliksel ve niteliksel ölçütlerdir. Aşağıda örnek bir projenin amacı ve hedefleri açıklanmaktadır.

Amaç

Bu projede reklam panolarına ticari amaçla reklam teklifinde bulunacaklara kolaylık sağlayacak çevrimiçi genel bir dijital reklam sisteminin geliştirilmesi amaçlanmaktadır.

Hedefler

- Kullanıcı dostu bir arayüz tasarlanması,
- Kullanıcıların uygun dış mekân ekranlarında görüntülenmek üzere video veya görsel yükleyebilecekleri web sitesinin oluşturulması,
- Android ve IOS tabanlı mobil uygulamanın oluşturulması,
- Reklam önerilerinin test edilebileceği bir ön dizinleme sisteminin geliştirilmesi,
- Reklam tekliflerinin onay süreçlerinin takibi ve sonucun bildirimi,
- Yayınlanmakta olan reklamlarla ilgili bilgi ve görüntülerin paylaşımı,
- Faturalandırma ve ödeme sistemlerinin geliştirilmesi ve takibi.

  ……..

2 Projenin Kapsamı (14 font)

Geliştirilecek yazılım ürünü ile ilgili olarak proje takımının yaptığı çalışmanın ilk çalışma olmadığı kabul edilmektedir. Piyasadaki güncel yazılım ürünlerinin pek çoğu da önceden geliştirilmiş ve daha sonra yeniden yapılandırılmış ürünlerdir. Yeni bir ürün geliştirirken iş analistleri eski ürün yerine yenisinin niçin geliştirilmesi ya da önceden elle yapılan birtakım işlerin neden otomatikleşmesi ya da değiştirilmesi gerektiğini araştırırlar. Proje konusu konu için de bu soruların cevabı ve niçin tercih edildiği araştırılmalıdır. Geliştirilecek ürün mevcut iken niçin yeniden geliştirilmesine gerek duyulmuştur? Diğer bir ifade ile mevcut sistemde ne gibi eksilikler görüldü ki ürünün yeniden geliştirilmesi istenmiştir? **şeklindeki sorular mutlaka cevaplanmalıdır**.

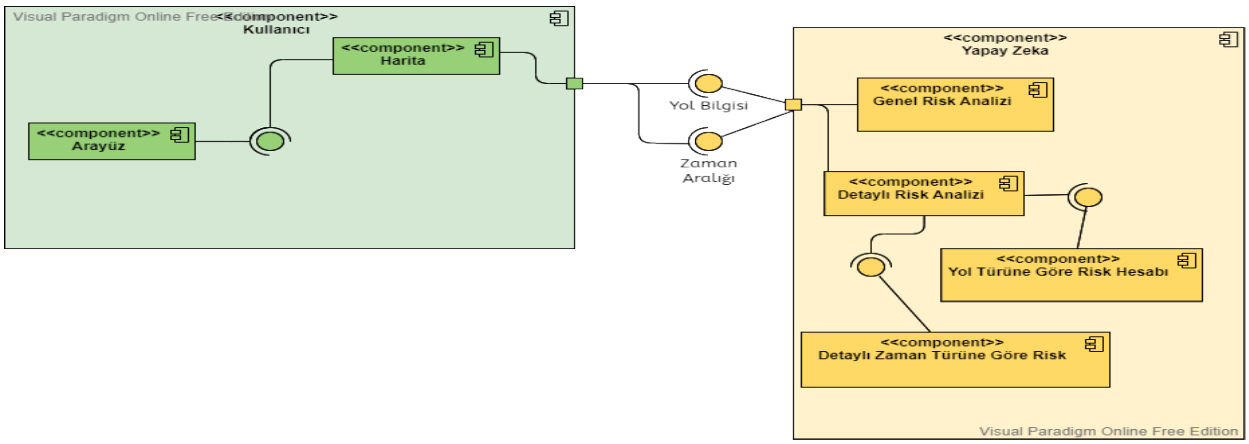
2.1 Projenin İçeriği

Proje çalışmasına ait "*component diagram*" ın çizilmesi istenmektedir. Böylece çalışmanın üst düzeydeki açıklaması görsel olarak yapılmış olacaktır. Diğer bir ifadeyle, "*component diyagram"* ile projenin sınırları belirlenecektir. Böylece, geliştirilecek ürün tüm alt bileşenleri ile birlikte açıklanmış olacaktır. Sadece UML diyagramının çizilmesi yeterli değildir, üst düzeydeki bu görsel çözümleme mutlaka ***yazılı olarak ta açıklanmalıdır.***

Çalışmada UML diyagramları çizilirken "visual paradigm community edition" kullanılması önerilir. (https://www.uml-diagrams.org/component-diagrams.html). Çevrimiçi erişilebilen UML linkleri tasarım projenizin diyagramlarını çizmek için yeterli olmayacaktır. UML geliştirme

araçları kullanılmadan çizilen görsel **çözümlemeler geçerli değildir**. Hangi UML görsel çözümleme aracı kullanıldığı **mutlaka belirtilmelidir.**

Çizdiğiniz tüm diyagramlar **şeklinizin altında** belirtilmelidir ve şekli metin olarak anlatırken atıf yapılmalıdır. Örneğin metin içerisinde "*Şekil 1'de olduğu gibi……. işlevleriyle tasarlanmıştır*" biçiminde açıklanmalıdır. Benzeri açıklamalar raporun **her bölümündeki şekil ve tablolar** için yapılmalıdır.



Şekil 1: ……..sistemine ait "component" diyagramı

3 Fonksiyonel Olmayan Gereksinimler

Fonksiyonel olmayan gereksinimler geliştirilen sistemin başarısının göstergesi olacağı için çevik ekipler için hayati beceridir ve sistemin niteliklerini belirtir. Bunlar, sistemin ne yaptığının değil, *onu ne kadar iyi yaptığın*ın göstergesidir. Dolayısıyla sistemin çeşitli girdilere yanıt olarak ne yapacağını tanımlayan fonksiyonel gereksinimlerden tamamen farklıdır. Fonksiyonel olmayan gereksinimlerin karşılanmaması, sistemin yapılan işi yani müşterilerin ihtiyaçlarını karşılamaması demektir; bu da uygulama geliştirilirken kullanılan standartların da karşılanmamış olması demektir. Örneğin uyumsuzluk (non-compliance); maliyet, geri çekme, gizlilik (privacy), güvenlik (security), güvenlik riski (safety risk), yasal yaptırımlar (legal exposure) gibi önemli sorunlara neden olabilir. O nedenle fonksiyonel olmayan gereksinimlerin doğru tanımlanması ve uygulanması kritik öneme sahiptir. Aşırı sayıda belirtim durumunda çözüm çok maliyetli veya uygulanamaz olabilir. Hedeflenen hiçbir gereksinim yoksa sistem kullanım açısından yetersiz olabilir. Sistemin kapsamı ne olursa olsun, fonksiyonel olmayan gereksinimler belirtilmelidir.

Çalışma grupları çözülecek probleme özgü fonksiyonel olmayan gereksinimlerini ve açıklamalını gerekçelerini de ekleyerek hazırlamalıdır. Fonksiyonel olmayan gereksinimlerin belirlenmesi ve açıklamaları için https://en.wikipedia.org/wiki/Non-functional_requirement linkini incelemeniz önerilir.

Fonksiyonel olmayan gereksinimlere birkaç örnek aşağıdadır:

…….

Güvenilirlik : ……., insan ve etkilenebilecek canlı cansız tüm varlıkların maddi manevi duyarlılıklarının korunmasını hedeflediğinden çalışma ortamı ve kriterleri dahilinde çevrelendiğinde %99,9 hatasız çalışması gereklidir.

Sürdürülebilirlik: …….sisteminin arıza vermesi durumunda bakım yapılabilmesi için o anda hiçbir kullanıcı tarafından kullanılmadığına emin olunmalı ve MTRS seviyesi olabildiğince az tutulmalıdır.

Kullanılabilirlik: Kullanıcının veri girişi yaparken internet bağlantısında bir sorun olmadığı kabul edildiğinde, verilerin başarılı analizi sonucunda sistem tüm kullanıcılar için 7/24 kullanılabilir durumda olmalıdır.

……..

II Gereksinimlerin Analizi (16 font) (Ana bölümler mutlaka yeni sayfadan başlamalıdır)

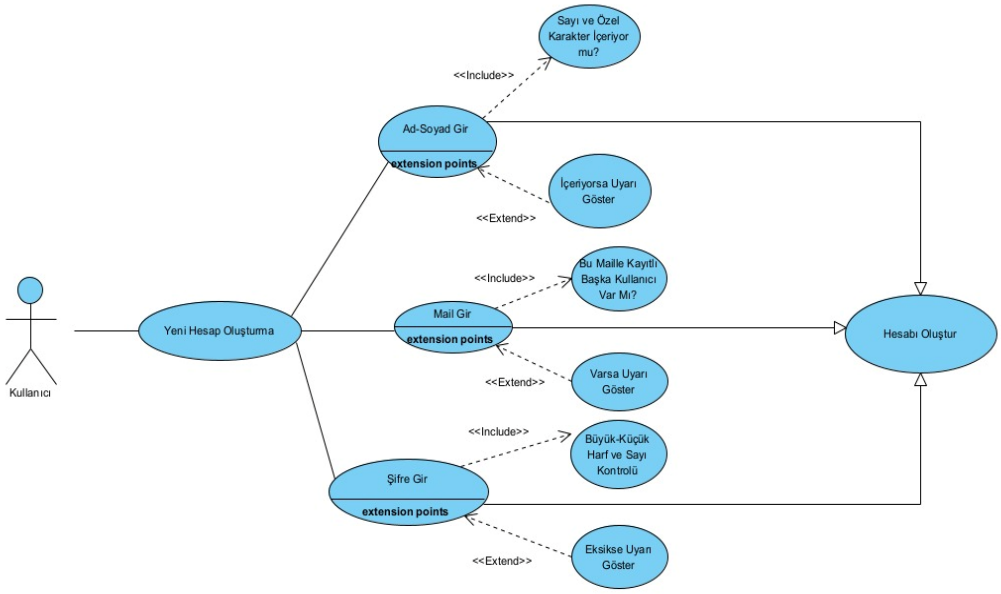
4 *Use Cases* ve Tablo Açıklamaları (14 font)

Uses cases, geliştirilecek yazılım sisteminin ve sistemin sınırlarının tanımlanmasıyla birlikte sistemin fonksiyonel gereksinimlerinin tanımlanmasını da sağlar. Use case diyagramları (https://www.uml-diagrams.org/use-case-diagrams.html) linki referans alınarak geliştirilebilir.

Bu bölümde tanımlanan *use case* diyagramları ve tablo üzerindeki açıklamaları, Rapor 2 olarak hazırlanacak tasarım aşamasına ait UML diyagramlarına değiştirilmeden yansıtılacaktır. Rapor 2 içerisinde bu rapordaki *use case diyagramları ile örtüşmeyen çözümler değerlendirilmeyecektir.*

Geliştirilecek yazılım ürününün gereksinimlerinin belirlenmesi ve her birinin tablo ile açıklanmasından sonra, sistemin sınıflarının tanımlanacağı ve aralarındaki ilişkilerin çıkartılacağı tasarım aşamasına geçilecektir.

Bir *use case* diyagramı ve bir başka örnek için use case diyagramının tablo üzerindeki açıklaması örneği aşağıdadır.



Şekil 2 : ………..'a ait use case diyagramı (Şekil bildirimi şeklin altında yapılır)

Tüm şekil ve tablolar metninizin bu bölümünde yapılan açıklamalar içerisinde mutlaka refere edilemelidir.

use cases sayısı, projedeki grup elemanlarının sayısına ve bu bağlamda gerçekleştirilecek projenin kapsamına göre belirlenmelidir ve de giriş /çıkış yapılması gibi basit çözümlemeler olmamalıdır. Bu diyagramlar mutlaka bahar dönemindeki Bitirme projesi dersinde implementasyonu gerçekleştireceğiniz karmaşık probleminizi açıklayacaktır.

Tablo1* : ……..use case diyagramı açıklaması (başlık mutlaka tablonun üstüne yazılır)

Use case	**SK.1.0**
Aktörler	Admin,Owner, Housekeeper
Ön Koşullar	Owner ya da Admin bulunmalıdır. Property belirlenmelidir
Son Durum	Housekeeper verilen görevi tamamlayacak, gerekli durumlarda Assets stoğu güncellenecektir.

Başarılı Senaryo	1. *Owner property* belirler, *admin* üyeliği oluşturur. 2. *Owner* ya da *Admin sector* ve *space* alanı belirler, *Housekeeper* üyelikleri oluşturur. 3. *Owner* ya da *Admin Task* bilgileri gönderir. 4. *Housekeeper Task*'ı alır, *Status inprogress* halini alır 5. *HouseKeeper Task*'ın durumuna göre *Status* tekrar düzenlenir. 6. *Task* tamamlanır, belirli durumlara göre *Assets* stoğu güncellenir.
Alternatif Senaryo	*Housekeeper Task* için *issued* durumu işaretlerse, gerekli *Assets* stok düzenlemeleri yapılır, *Task*'ın *done* olması sağlanır

*Tablo 1 'de olduğu gibi İngilizce sözcükler çalışma alanına **özel terimler** olarak kullanılıyorsa Türkçe karşılığı ile yazılmak zorunda değildir. Böylece kavramsal çözümün okunabilirliği kolaylaşır.

Her *use case* diyagramının bir listesini yapmak, her birini ayrı ayrı modellemek ve tanımlamak proje çalışmasının analiz aşamasının izlenebilirliğini kolaylaştıracaktır. Bu aşamada çalışma grubunun her üyesinin aktif olması ve çözüm tartışmalarına katılması çalışmanın kalitesinde önemlidir.

Use case diyagramlarının sistem üzerinde iki amacı vardır: Her bir *use case* diyagramının problemin ilgili parçasına ait kavramsal çözümlemede *include* ve/veya *extend* ilişkileri tanımlanmalıdır. Bu ilişkiler karmaşık bir problemin çözümüne doğal olarak bulunur. Her bir *use case* diyagramını listeleyen tablo, bu *use case* diyagramının sistemin bir parçası olarak hangi eylemleri içerdiğini ve de aynı zamanda sınırlamaları (kısıtları) ifade ederek sistemin neleri içermediğini gösterir.

Aktörlerin canlı ve/veya cansız bir varlık olması mümkündür. Çözülen problemin karmaşıklığına bağlı olarak, bazı *use case* diyagramları için birden fazla diyagram kullanılması gerekebilir.

Proje grubu geliştirilecek yazılım ürününün sınırlarına karar vererek ürünün use *cases* tasarlar ve probleminin gereksinimler analizini tamamlar. Bu sınırın ne olacağı (çözümün ne büyüklükte olacağı, diyagramların sayısı ve birbirlerine bağlılığı) grubun çalışma konusundaki bilgisine, gayretine ve gereksinim kısıtlamalarına göre belirlenecektir.

5 Fonksiyonel Gereksinimler

Her bir olay (vaka) ya da diğer bir ifade ile *use case* kullanımı listesi için fonksiyonel gereksinimler aşağıdaki gibi bir tabloya yazılacaktır. Her bir fonksiyonel gereksinim için tek (*unique)* bir sayı belirlenecektir. Bu bölüm çalışmanın izlenebilirliğini (tracebility) ve geliştirme aşamasını kolaylaştırır.

Tablo 2: Problem Gereksinimlerinin Tanımı

Gereksinim ID	**Gereksinim Tanımı**
SK.1.0	use case ile sisteme kayıt işlemleri yapılır.
DK.1.0	use case, işlemler sırasında istenilen doğrulama kodlarını gönderir.
SG.1.0	use case ile sisteme giriş işlemleri yapılır..
SU.1.0	use case, şifre unutma durumunda yeni şifre alınmasını sağlar.
HI.1.0	use case, hesaplama işlemlerinin yapılmasını sağlar.
EI.1.0	use case, editöre ait işlemlerin yapılmasını sağlar.
SYI.1.0	use case, ana sistem yönetimine ait işlemlerin kontrolünü sağlar.
KK.1.0	use case, sisteme kayıp varlığın kaydının yapılmasını sağlar.
IA.1.0	use case kullanıcın ilan arama işlemlerini yapmasını sağlar.
DA.1.0	use case kullanıcının depo arama işlemlerini yapmasını sağlar.
IA.1.1	use case, listelenen ilanlar arasında kullanıcın seçmiş olduğu ilanın bilgilerini gösterir.
DA.1.1	use case, listelenen depolar arasında kullanıcın seçmiş olduğu deponun bilgilerini gösterir.

ÖNEMLİ NOT: Doküman içerisindeki örnek şekil ve tabloların her biri diğeri ile bağlantılı olmayacak şekilde seçilmiştir.