BLM3722

Yazılım Mühendisliği

GİDER YÖNETİM SİSTEMİ

5.GRUP

23011602 - Ebru Kılıç

22011647 - Sinem Sarak

21011037 - İclal Ertürk

22011934 - Meryem İbrahimoğlu

Prof. Dr. Oya Kalıpsız

Mayıs,2025

İÇİNDEKİLER

1. Proje Alan Tanımı: Gider Yönetim Sistemi

1.1. Proje Tanımı ve Amacı

Gider Yönetim Sistemi, şirketlerin personellerinin iş amaçlı yaptığı sahadaki harcamalarını izlemek ve bu harcamaların tazmin süreçlerini yönetmek için tasarlanmış bir bilgi sistemidir. Bu projede şirketler, personelinin yaptığı harcamaları doğru bir şekilde takip edebilecektir ve bu harcamalar onaylandıktan sonra muhasebe birimi tarafından tazmin işlemi gerçekleştirilebilecektir. Ayrıca, sistem, harcama kalemlerine ilişkin sınırları ve bütçeleri kontrol edebilecek ve bu veriler ışığında gelecek yıla yönelik bütçe tahminleri yapabilecektir. Sistemin temel işlevleri arasında birim ve kalem bazında bütçe ve limitlerin belirlenmesi, yapılan harcamaların kaydedilmesi, onaylanması, tazmin edilmesi ve geçmiş verilere dayalı olarak geleceğe yönelik bütçe tahminleri yapılması bulunmaktadır.

1.1.2. Proje Kapsamı

Sistem, aşağıdaki temel özelliklere sahip olacaktır:

Harcama Kaydı: Çalışan üst yönetici tarafında belirlenen kalemler bazında yaptığı harcamaları sisteme kaydeder. Örneğin, taksi ücreti, otopark bedeli, benzin gibi masraflarını sisteme girmesi gerekmektedir.
Birim Yöneticisi Onayı: Harcama kaydı, çalışan tarafından sisteme girildikten sonra, çalışanın bağlı olduğu birim yöneticisi tarafından onaylanacaktır.
Tazmin Süreci: Onaylanan harcama belgeleriyle birlikte, çalışan muhasebe birimine başvuracak ve harcama tazmin edilecektir. Muhasebe birimi çalışanların harcamalarını listeleyebilmektedir. Belgenin teslimi konusunda sözleşme oluşturulmuştur. 4. bölüm 1. kısımda bahsedilen sözleşme yer almaktadır.
Bütçe Yönetimi: Şirketin her biriminin farklı harcama bütçeleri ve eşik değerleri olacaktır. Üst yönetim tarafından belirlenen limit aşımına dikkat edilerek işlemler yapılacak, limit aşıldığında belirlenen eşik değeri kadar ödeme yapılacaktır.
Raporlama: Yapılan harcamalar, çalışan, birim, harcama kalemi ve tarih bazında grafikler aracılığıyla görselleştirilecektir.
Bütçe Tahmini: Geçmiş yıllara ait harcama verilerine dayalı olarak, sistem gelecek yıl için bütçe önerisi sunacaktır. Bu, Hareketli Ortalama Yöntemi ile yapılacaktır.

1.2. Kabul ve Kısıtlar

Bu proje, gider yönetimi sürecini daha verimli ve kontrollü bir şekilde dijitalleştirmeyi amaçlamakta olup belirli kısıtlamalar altında ve önceden belirlenmiş gereksinimler doğrultusunda gerçekleştirilecektir. Bu kısıtlamalar, projeyi belirli bir çerçevede tutarak yönetilebilir ve güvenilir bir sistem geliştirilmesine olanak tanıyacaktır.

1.2.1. Kabul Edilen Özellikler

Bu projede aşağıdaki özellikler kabul edilmiştir:

Harcama Kaydı ve Onayı: Çalışanlar sahada gerçekleştirdiği harcamaları sisteme kaydedebilecek, bu kayıtlar birim yöneticileri tarafından onaylanacak, ardından muhasebe birimi tarafından tazmin edilecektir.
Birim Yöneticisi Onayı ve Tazmin Süreci: Harcama kaydını onaylayan birim yöneticisi, harcamanın doğru kaleme yapılması gerektiğini kontrol eder. Onaylanan harcamalar, muhasebe birimi tarafından tazmin edilecektir.
Bütçe Yönetimi: Her birim için bütçe limitleri üst yönetim tarafından belirlenmiştir. Limit aşımı durumunda, üst yönetim tarafından belirlenen eşik değerine kadar ödeme yapılacaktır. Ödeme işlemlerinin kontrolü, tazmin süreci, muhasebe birimi tarafından gerçekleştirilecektir.
Hareketli Ortalama Yöntemi: Geçmiş verilere dayalı olarak, sistem gelecek yıl için bütçe önerisi yapmak amacıyla Hareketli Ortalama Yöntemini kullanacaktır.
Raporlama ve Analiz: Yapılan harcamalar, kişi, birim, kalem ve tarih bazında görselleştirilecektir. Bu görselleştirme, harcama analizlerinin doğru yapılmasını ve karar vericilerin gerekli bilgiyi grafik üzerinde görerek anlamlandırmasını sağlayacaktır.

1.2.2. Kısıtlamalar

Projede aşağıdaki kısıtlamalar geçerli olacaktır:

- **Veri tabanı Seçimi**: Veri tabanı olarak SQLite kullanılacaktır.
- **Bütçe Tahmin Yöntemi**: Hareketli Ortalama Yöntemi dışında başka herhangi bir tahminleme algoritması kullanılmayacaktır. Karmaşık modeller veya yapay zekâ tabanlı tahminler sisteme dahil edilmeyecektir.
- **Muhasebe Birimi Harcama Bilgisi:** Harcama Belgesi çalışan ekranında oluşturulmaktadır. Çalışan gerektiği durumda bu belgeyi muhasebe birimine elden teslim edecektir.
- **Zaman Kısıtlaması**: Proje, belirlenen zaman diliminde tamamlanacaktır.
- **Bütçe Kısıtlaması**: Proje, belirli bir bütçeye sahiptir. Bu nedenle, ilave yazılım veya donanım kaynaklarına harcama yapılmayacaktır.

1.2.3. Varsayımlar

Harcama Kalemleri: Sistem, önceden tanımlanmış harcama kalemlerini kullanacaktır. Yeni kalemler eklenmeyecek ve mevcut kalemler üzerinden harcama yapılacaktır.
Çalışanın Belgesi: Çalışan belge çıktısını alacak ve muhasebeye onaylatmak için elden teslim edecektir.
Bütçe Limitleri: Harcama limitleri ve eşik değerleri sabit olacak ve yalnızca üst yönetim tarafından değiştirilebilecektir. Diğer roldeki çalışanlar bu değerleri değiştiremeyecektir.

1.3. Risk Tablosu

Proje sürecinde karşılaşılabilecek potansiyel riskler önceden belirlenmiş, bu risklerin oluşma olasılıkları ile projeye olan etkileri analiz edilerek risk skoru hesaplanmıştır. Her bir risk için alınacak önlemler belirlenerek aksiyon planları oluşturulmuştur. Bu yaklaşım, projenin hem sürdürülebilirliğini hem de başarısını güvence altına almak amacıyla uygulanmıştır.

Aşağıda yer alan tablo, bu analiz doğrultusunda belirlenen başlıca riskleri ve bunlara karşı planlanan aksiyonları göstermektedir:

Tablo 1.1 Risk Tablosu

Risk No	Risk Tanımı	Olasılık (1-3)	Etki (1-3)	Risk Skoru (O×E)	Alınacak Önlem / Aksiyon Planı
R1	Kullanıcı rolü hatalı algılanır	1	3	3	Giriş sonrası rol kontrolü yapılır, backend'de doğrulama eklenir.
R2	Harcama talepleri statüde kalır	2	3	6	Durum değişikliği sonrası tablo otomatik olarak yeniden yüklenir.
R3	Arayüzde kolonlar görünmez veya taşar	3	2	6	Otomatik kolon genişliği ve responsive tasarım uygulanır.
R4	OOP kurallarına uyulmaması	2	2	4	Kod sınıflara bölünür, yeniden yapılandırma (refactor) yapılır.
R5	Test kapsamı yetersiz	2	2	4	Tüm kritik fonksiyonlar için birim testleri yazılır.
R6	Test verisi eksik veya hatalı	1	3	3	Test veri seti kontrol edilir, veritabanına örnek kayıtlar girilir.
R7	PDF raporu üretilemiyor	2	2	4	Rapor üretiminde hata yakalanır, kullanıcıya uyarı verilir, log tutulur.
R8	Tazmin aşımı limit kontrolü yanlış çalışıyor	2	3	6	Limit ve oran kontrolleri için test senaryoları eklenir, log ile takip yapılır.
R9	UI öğeleri cihaz çözünürlüğüne göre bozuluyor	3	2	6	Qt arayüz responsive olacak şekilde esnek yapılandırılır.
R10	Kullanıcı çıkışında oturum tam kapanmıyor	1	3	3	close() ve show() metodları dikkatle kontrol edilir, kaynaklar serbest bırakılır.
R11	PDF yolunun veri tabanına yazılamaması	2	2	4	PDF üretiminden sonra güncelleme işlemi try-except ile korunur.
R12	Aynı harcama birden fazla kez işleniyor	1	3	3	Her harcama için benzersiz ID kontrolü yapılır.

- **Olasılık:** Riskin gerçekleşme olasılığı (1: düşük, 3: yüksek)
- **Etki:** Gerçekleştiğinde projeye etkisi (1: düşük, 3: yüksek)
- **Risk Skoru:** Olasılık × Etki
- **Önlem:** Riskin etkisini azaltacak stratejiler

Bu tablo sayesinde, proje sürecinde oluşabilecek aksaklıklar proaktif olarak ele alınmış ve risklerin en aza indirilmesi hedeflenmiştir.

1.4 İzlenebilirlik Matrisi

Projenin gerçekleştirilebilmesi için müşteri tarafından sağlanan gereksinim metni, tüm ekip tarafından analiz edilmiştir. Bu analiz sürecinde, sistemin işleyişine dair ihtiyaçlar belirlenmiş, eksik ya da belirsiz noktalar; varsayım ve sözleşmelerle netleştirilmiştir. Elde edilen bilgiler doğrultusunda sistemin işlevsel ve işlevsel olmayan gereksinimleri ayrıştırılmış, her bir gereksinimin hangi kullanıcı rolü tarafından karşılanacağı ve sistemin hangi bileşenleriyle ilişkili olacağı belirlenmiştir.

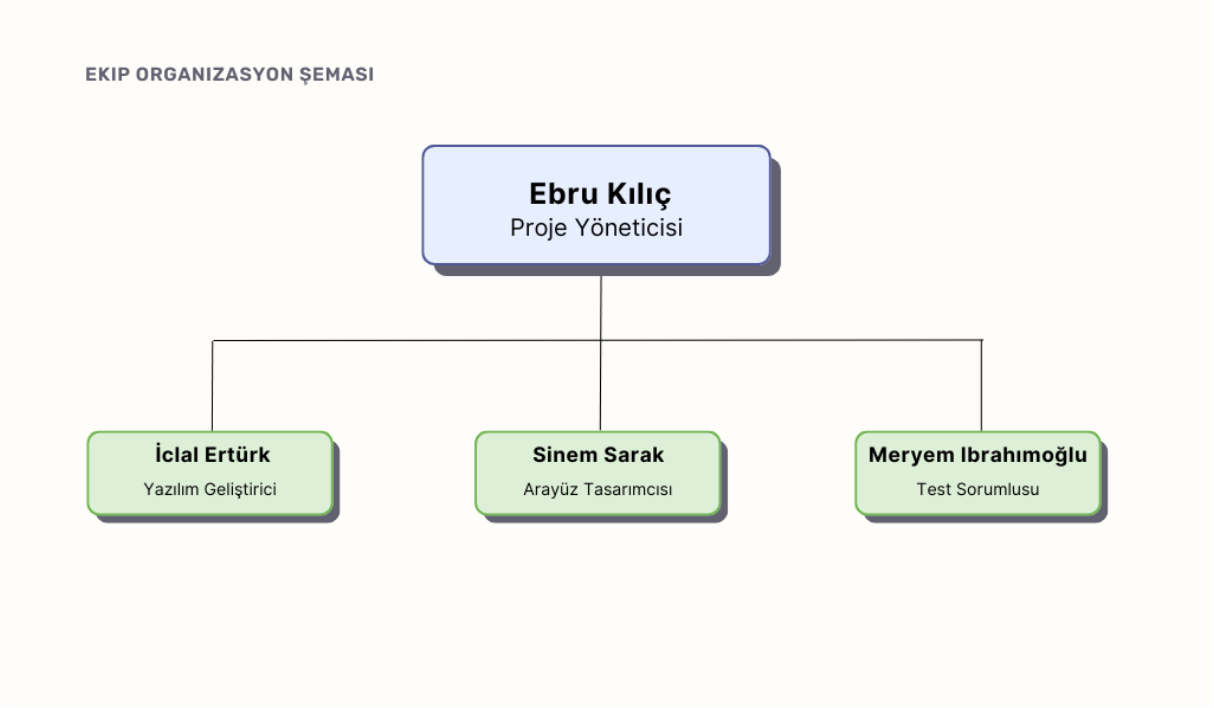
Bu kapsamda, temel gereksinimler başlıklar altında sınıflandırılmış; her bir gereksinimi karşılayacak olan modüller, sınıflar ve kullanıcı arayüzleri planlanarak sistemin genel mimarisi oluşturulmuştur. Ayrıca bu gereksinimlerin izlenebilirliğini sağlamak amacıyla, kullanıcı rolleri ile gereksinimler arasındaki ilişkiyi gösteren izlenebilirlik tablosu hazırlanmıştır. İzlenebilirlik tablosu Tablo 1.2' de verilmiştir.

Tablo 1.2: İzlenebilirlik Matrisi

		employee sınıfı	yonetici sınıfı	muhasebe sınıfı	budget_manager sınıfı	tahmin sınıfı
Çalışan Rolü	G1 - Çalışan birimine tanımlı olan harcama kalemlerini görüntüleyebilmelidir.	✓				
	G2 - Çalışan istediği kaleme dair harcama girebilmelidir.	✓				
	G3 - Çalışan oluşturmuş olduğu harcama taleplerini, onay durumlarını ve tazmin miktarlarını bir liste halinde görebilmelidir.	✓				
	G4 - Çalışan onaylanan harcamalarına dair detay belgesi oluşturulup indirilebilmelidir.	✓				
Yönetici rolü	G5 - Yönetici sadece kendi birimine dair harcama taleplerini bir liste halinde görüntüleyebilmelidir.		✓			
	G6 - Yönetici harcama talepleri listesindeki talepleri onay durumlarına göre filtreleyebilmelidir.		✓			
	G7 - Yönetici harcamaları onaylayabilmeli veya reddedebilmelidir.		✓			
Muhasebe Rolü	G8 - Muhasebe çalışanı yanlış girilmiş bir harcama kalemini düzeltebilmelidir.			✓		
	G9 - Muhasebe çalışanı bekleyen tazmin taleplerini bir lise halinde görüntüleyebilmelidir.			✓		
	G10 - Muhasebe çalışanı seçtiği harcamanın ait olduğu birimin bütçe detaylarını görüntüleyebilmelidir.			✓		
	G11 - Muhasebe çalışanı seçtiği harcama için tazmin bedelini girebilmelidir.			✓		
Üst yönetici Rolü	G12 - Üst yönetici şirketteki tüm departmanların bütçe detaylarını bir liste halinde görüntüleyebilmelidir				✓	
	G13 - Üst yönetici şirkette istediği departmana yeni bütçe tanımlayabilmelidir.				✓	
	G14 - Üst yönetici tanımlanmış olan her birim ve kalem için ayrı bütçe limiti belirleyebilmelidir.				✓	
	G15 - Üst yönetici her birim ve kalem için ayrı aşım oranı belirleyebilmelidir.				✓	
	G16 - Üst yönetici tüm çalışanları görüntüleyebilmelidir.				✓	
	G17 - Üst yönetici seçtiği birimin tüm birimlerin harcamaları içerisindeki dağılımını grafik halinde görüntüleyebilmelidir				✓	
	G18 - Üst yönetici seçtiği kalemin birim bazındaki harcama dağılımını grafik halinde görebilmelidir.				✓	
	G19 - Üst yönetici seçtiği personelin kendi departmanı içerisinde bulunan diğer personellerin harcamaları içerisindeki dağılımını grafik halinde görüntüleyebilmelidir.				✓	
	G20 - Üst yönetici seçtiği periyoda göre gelecek yıl için tahmin alabilmeli ve hareketli ortalama kullanılarak oluşturulan grafiği görebilmelidir.					✓

2. Proje Ekip Yapısı ve Organizasyon Şeması

Projenin sağlıklı bir şekilde yürütülebilmesi için ekip üyeleri arasında açık bir rol dağılımı yapılmış ve herkesin sorumluluk alanı netleştirilmiştir. Aşağıda yer alan organizasyon şeması, proje ekibinin yapısını göstermektedir:



Görsel 2.1 - Ekip Organizasyon Şeması

- **Ebru Kılıç** proje yöneticisi olarak genel koordinasyonu sağlamıştır Ayrıca teknik fizibilite, veri tabanı tasarımı ve üst yönetici arayüzü, fonksiyonları ile diyagramlarında aktif rol almıştır.
- **İclal Ertürk** yazılım geliştirici olarak hem kod geliştirme süreçlerinde (muhasebe ve üst yönetim modülleri) aktif rol almış hem de UML sınıf diyagramı, muhasebe diyagramları, sosyal ve yasal fizibilite, Gantt diyagramı gerçekleştirmede aktif rol almıştır.
- **Sinem Sarak** arayüz tasarımından sorumlu olup, özellikle çalışan paneli arayüzü, modülleri ve ilgili diyagramları üstlenmiştir. Ayrıca durum diyagramı ve izlenebilirlik tablolarını da gerçekleştirmiştir.
- **Meryem İbrahimoğlu** test sorumlusu olarak hem test süreçlerini yürütmüş hem de yönetici modülünün tasarımı, fonksiyonları ve risk değerlendirmesinde görev almıştır.

Tablo 2.1 Görev Dağılımı Özeti Tablosu

Görev	Sorumlu
Kabul ve kısıtlar	Tüm Ekip
Gantt diyagramı	İclal
Risk tablosu	Meryem
Teknik fizibilite	Ebru
Ekonomik fizibilite	Sinem

Zaman ve iş gücü fizibilitesi	Tüm Ekip
Yasal ve sosyal fizibilite	İclal
Yönetim fizibilitesi	Meryem
Use-case diyagramı	Ebru
UML sınıf diyagramı	İclal
İzlenebilirlik tablosu	Sinem
State diyagramı	Sinem
Veri tabanı tasarımı	Ebru
Çalışan modülü & testleri	Sinem
Üst yönetici modülü & testleri	İclal, Ebru
Yönetici modülü & testleri	Meryem
Muhasebe modülü & testleri	İclal

Bu görev paylaşımı sayesinde ekip, proje boyunca iş bölümü içinde çalışmış ve proje hedeflerine zamanında ulaşmayı başarmıştır.

3. Sistem Analizi

3.1. Teknik Fizibilite

Bu bölümde, projede kullanılacak teknolojiler, sistem gereksinimleri ve entegrasyon süreçleri teknik açıdan değerlendirilmektedir. Seçilen işletim sistemi, programlama dili, veri tabanı yönetim sistemi ve kullanıcı arayüzü için kullanılacak framework gibi teknik gereksinimler açıklanmaktadır.

İşletim sistemi olarak entegrasyon kolaylığı, geniş geliştirme araçları, kullanıcı dostu arayüz, kurumsal destek ve uygulama uyumluluğu sağlamasından dolayı projenin geliştirilme ve çalıştırılma süreci için en uygun olacağı düşünülen Windows işletim sistemi tercih edilmiştir. Yazılım dili olarak Python kullanılacaktır. Python programlama dilinin kolay okunabilirliği, geniş kütüphane desteği, nesne yönelimli programlama paradigmasını desteklemesi ve çok platformlu çalışabilme yeteneği projenin tüm gereksinimlerini karşılamaktadır. Yazılım geliştirme sürecinde Python programlama dilinin sunduğu geniş ekosistem ve güçlü framework desteği, sistemin esnek ve ölçeklenebilir bir yapıya sahip olmasına katkı sağlayacaktır.

Python’da veri tabanı işlemleri için pek çok alternatif bulunmaktadır. Bu alternatifler içerisinde projede SQLite kullanılmasına karar verilmiştir. SQLite herhangi bir yazılım veya sunucu kurulumu gerektirmemektedir. Bu sayede, bu modülü kullanabilmek için öncelikle bir sunucu yapılandırılmasına gerek yoktur ve ücretsizdir. SQLite, düşük kaynak tüketimiyle yeterli hız ve verimlilik sunar. Küçük ölçekli projelerde yeterli performans sağladığından, ek bir veri tabanı yönetim sistemine ihtiyaç duyulmamaktadır. Bu sebeplerden ötürü SQLite veri tabanı yazılımının proje için en uygun seçenek olduğuna karar verilmiştir.

Projenin kullanıcı arayüzü tasarımı için PyQt framework'ü kullanılacaktır. PyQt, masaüstü uygulamaları için kullanılan grafik kullanıcı arayüzü geliştirme ortamıdır. Geniş Qt kütüphaneleri, proje kapsamında kullanıcı deneyimini geliştirecek çeşitli özelliklerin sisteme entegre edilmesine ve kullanıcı dostu arayüzün geliştirilmesine olanak tanıyacaktır.

Projenin yazılım gereksinimlerini karşılayabilecek, gelişime açık bir sistem oluşturmak önemlidir. Uygulamanın minimum donanım gereksinimleri Tablo 3.1'deki gibidir.

Tablo 3.1 Minimum Donanım Gereksinimleri

Donanım Gereksinimleri	
İşletim Sistemi	Windows 10 veya üstü
İşlemci	Intel i5/AMD Ryzen 5 veya üstü
RAM	8 GB veya üstü
Depolama	Minimum 10 GB boş alan
Ağ Bağlantısı	Wireless Router

3.2. Ekonomik Fizibilite

Projenin ana maliyet kalemleri aşağıdaki gibi belirlenmiştir:

Geliştirme ve Test Ortamı:

- Veri tabanı ve sunucu olarak Python'un Django framework'u kapsamında ücretsiz olarak bulunan Sqlite kullanılacak, proje localhost üzerinde çalıştırılacaktır. Dolayısıyla veri tabanı ve sunucu maliyeti bulunmamaktadır.
- Python dili ve Python diliyle geliştirilmiş açık kaynak kütüphaneler kullanılacağından yazılım lisans maliyeti bulunmamaktadır.
- Geliştirmenin yapılacağı Visual Studio Code ücretsiz olduğundan maliyeti bulunmamaktadır.
- Geliştirme Windows 11 üzerinde yapılacaktır. Windows lisansları üniversite tarafından sağlandığı için ek maliyet yoktur.

Donanım Maliyeti:

Projenin maliyet analizi yapılırken donanım ekipmanları için amortisman hesaplaması yapılmıştır. Kullanılan ekipmanların maliyetleri ve kullanım süreleri göz önüne alınarak, doğrusal amortisman yöntemi kullanılmıştır. Bu yöntem kapsamında mevcut teknolojik cihazların ömrü Moore yasasına uygun olacak şekilde 18 ay olarak alınmış, proje süreci 2 ay olarak hesaplanmıştır. Hesaplanan maliyetin detayları Tablo 3.2'de verilmiştir.

Tablo 3.2 Hesaplanan Donanım Maliyeti

Cihaz	Güncel Fiyat (TL)	Aylık Amortisman (TL)	2 Aylık Amortisman (TL)
Bilgisayar 1	25.000	1.389	2.778
Bilgisayar 2	38.000	2.111	4.222
Bilgisayar 3	40.000	2.222	4.444
Bilgisayar 4	25.000	1.389	2.778
Bilgisayar 5	33.000	1.833	3.667
Toplam	17.889 TL		

Hesaplanan fizibilite doğrultusunda, projenin ekonomik açıdan uygulanabilir olduğu ve donanım maliyetlerinin sürdürülebilir seviyede olduğu sonucuna varılmıştır.

3.3. Yasal Fizibilite

Gider Yönetim Sistemi, çalışanların iş harcamalarını kayıt altına almak, onaylamak ve tazmin süreçlerini yönetmek için geliştirilmiştir. Sistem, Vergi Usul Kanunu ve Türk Ticaret Kanunu'na uygun olmalı, muhasebeleştirme süreçlerini düzenlemelidir. Kişisel verilerin korunması açısından KVKK gereklilikleri sağlanmalı, yetkisiz erişim engellenmelidir. İş Kanunu çerçevesinde çalışan hakları korunmalı, tazmin süreçleri şeffaf şekilde yürütülmelidir.

Veri güvenliği için erişim yetkilendirme ve güvenlik politikaları uygulanmalı, muhasebe kayıtları düzenli tutulmalı ve yetkilendirme mekanizmaları devreye alınmalıdır. Harcamalar, yasal çerçevede denetlenebilir olmalı, denetim ve raporlama süreçleri sağlıklı işleyerek şeffaf bir yapı oluşturulmalıdır. KVKK ihlallerinin önüne geçmek için güvenlik önlemleri alınmalı, vergisel usulsüzlükleri önlemek adına kayıt süreçleri sıkı şekilde takip edilmelidir. Siber güvenlik tehditlerine karşı sistem, ISO 27001 standartlarına uygun şekilde korunmalıdır.

Sonuç olarak, sistem yasal düzenlemelere tam uyumlu olmalı, denetim mekanizmaları oluşturulmalı ve veri güvenliği sağlanmalıdır.

3.4. Sosyal Fizibilite

Gider Yönetim Sistemi, çalışanların iş harcamalarını kolayca yönetmesini sağlayarak mali süreçleri şeffaflaştırır. Harcamaların dijital ortamda takip edilmesi, çalışanların gereksiz zaman kaybını önler ve işlemleri daha hızlı tamamlamalarına olanak tanır. Bu durum çalışan motivasyonunu artırarak iş süreçlerine odaklanmalarını sağlar.

Dijital süreçlerin kullanımı sayesinde kâğıt tüketimi azalır, bu da çevresel sürdürülebilirliğe katkı sağlar. Çalışanların tazmin taleplerinin hızlı ve etkin bir şekilde karşılanması, şirkete duyulan güveni artırırken, iş verimliliğini de yükseltir. Aynı zamanda, geçmiş yıllara ait verilerin analiz edilerek bütçe tahminleri yapılması, finansal planlamayı destekleyerek şirketin uzun vadeli stratejik hedeflerine ulaşmasına yardımcı olur.

Sonuç olarak, sistem çalışan deneyimini iyileştiren, güven ortamını artıran, çevresel sürdürülebilirliği destekleyen ve kurumsal yönetim süreçlerini güçlendiren bir yapıda olmalıdır.

3.5. Yönetim ve Zaman Fizibilitesi

Şirketin mevcut gider yönetim sürecinin manuel yürütülmesi; verilerin dağınık olması, onay sürecinde zaman kayıplarının yaşanması ve geçmiş harcamaların analizinde zorluklar oluşturması nedeniyle, yönetim yeni bir masaüstü yazılım uygulamasına geçme kararı almıştır. Bu doğrultuda tarafımıza ulaşılarak dijital bir gider yönetim sisteminin geliştirilmesi talep edilmiştir. Hazırlanan uygulamanın amacı; çalışanların harcamalarını kolayca sisteme girebilmesi, yöneticilerin bu harcamaları sistem üzerinden hızlı şekilde onaylayabilmesi ve muhasebe biriminin raporlara doğrudan erişerek süreci daha verimli yönetmesidir.

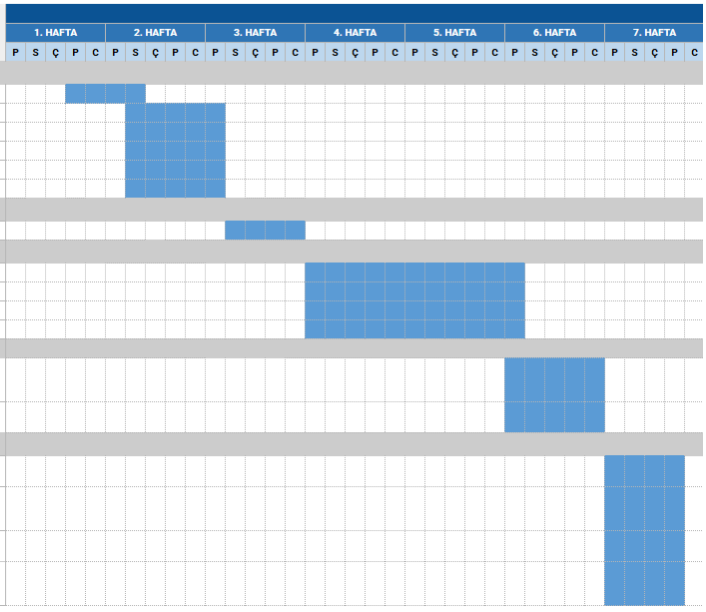
Analiz, tasarım ve geliştirme sürecinin tamamında yönetimle sürekli iletişimde kalınmıştır. Her aşamanın sonunda yapılan çalışmalar düzenli olarak raporlanmış, alınan geri bildirimler ışığında güncellemeler yapılarak bir sonraki aşamaya geçilmiştir. Böylece yönetimin istek ve ihtiyaçlarına doğrudan cevap verebilen, kullanıcı dostu bir sistem ortaya çıkarılmıştır.

Proje ekibinde çeşitli görev dağılımları yapılmıştır. Sistem analisti, kullanıcı gereksinimlerini analiz ederek projenin doğru temeller üzerine kurulmasını sağlamıştır. Tasarımcı, masaüstü arayüzün kullanıcı odaklı, sade ve işlevsel olmasına özen göstererek tüm ekranları tasarlamıştır. Geliştiriciler, sistemin arka plan mantığını ve veri tabanı işlemlerini kodlamıştır. Test uzmanı, tüm modüller için test senaryoları oluşturarak sistemin hatasız şekilde çalışmasını sağlamıştır. Destek personeli, sistem tesliminden sonra teknik sorunlara müdahale etmek ve kullanıcıya destek sağlamak üzere yapılandırılmıştır. Proje yöneticisi, tüm sürecin koordinasyonunu üstlenmiş, zaman yönetimi ve görev paylaşımını gerçekleştirmiştir.

GANTT ŞEMASI

PROJE BAŞLIĞI GİDER YÖNETİM SİSTEMİ **ŞİRKET ADI** STARTECH

WBS NUMARASI	GÖREV BAŞLIĞI	GÖREV SAHİBİ	BAŞLANGIÇ TARİHİ	BİTİŞ TARİHİ	SÜRE
1	**Proje Tanımı ve Gereksinim Analizi - Fizibilite Adımları**				
1.1	Ekip Toplantısı, Görev Dağılımı	Tüm Ekip Üyeleri	03.04.25	08.04.25	5
1.2	Teknik Fizibilite	Ebru	08.04.25	14.04.25	6
1.3	Ekonomik Fizibilite	Sinem	08.04.25	14.04.25	6
1.4	İş Gücü ve Zaman Fizibilitesi	Tüm Ekip Üyeleri	08.04.25	14.04.25	6
1.5	Yasal ve Sosyal Fizibilite	İclal	08.04.25	14.04.25	6
1.6	Donanım Fizibilitesi	Mariam	08.04.25	14.04.25	6
2	**Veri Tabanı Tasarımı - SQLITE**				
2.1	İlişkisel Veri Tabanı	Ebru	15.04.25	18.04.25	3
3	**Nesneye Dayalı Modelleme ve Tasarım - PYQT5**				
3.1	Çalışan Arayüzü ve İşlemleri	Sinem	21.04.25	05.05.25	14
3.2	Yönetici Arayüzü ve İşlemleri	Mariam	21.04.25	05.05.25	14
3.3	Üst Yönetim Arayüzü ve İşlemleri	İclal & Ebru	21.04.25	05.05.25	14
3.4	Muhasebe Arayüzü ve İşlemleri	İclal	21.04.25	05.05.25	14
4	**Test ve Hata Ayıklama**				
4.1	Uygulama Fonksiyonlarının Test Edilmesi ve Gerekli Güncellemelerin Yapılması	Ebru & İclal	05.05.25	09.05.25	4
4.2	Arayüz Şemalarının Gözden Geçirilmesi ve İyileştirilmesi	Sinem & Mariam	05.05.25	09.05.25	4
5	**Dağıtım&Kurulum- Proje Gerçeklemesi**				
5.1	Çalışan modülünün (profil, harcama) kurulumu ve son kontrolleri	Tüm Ekip Üyeleri	12.05.25	15.05.25	3
5.2	Yönetici modülünün (onay, bütçe, raporlama) kurulumu ve test sonrası uygulanması	Tüm Ekip Üyeleri	12.05.25	15.05.25	3
5.3	Üst yönetim panelinin (tahminleme, eşik belirleme) sistemle entegrasyonu	Tüm Ekip Üyeleri	12.05.25	15.05.25	3
5.4	Muhasebe panelinin kurulumunun yapılması ve tazminat işlemlerinin denenmesi	Tüm Ekip Üyeleri	12.05.25	15.05.25	3



Görsel 3.5.1 - Gantt Diyagramı

Proje süreci, çözümleme → tasarım → gerçekleme → sınama → teslimat & bakım adımlarına ayrılmıştır. Her aşama için süreler ve sorumlu kişiler belirlenmiş, süreç Gantt şeması (Görsel 1) ile planlanmıştır. Masaüstü yapıda geliştirilen sistemde Python programlama dili tercih edilmiş, arayüz için pyqt kütüphanesi, veri tabanı yönetimi için ise yerel uygulamalarla uyumlu, kuruluma ihtiyaç duymayan SQLite veri tabanı kullanılmıştır.

Tüm bu bilgiler doğrultusunda; görev paylaşımının açıkça yapılmış olması, kullanılacak kaynaklara erişimin kolaylığı ve yönetimin süreç boyunca aktif rol oynaması sayesinde projenin yönetimsel açıdan uygulanabilirliği yüksek düzeydedir. Projenin zamanında ve istenilen nitelikte tamamlanması öngörülmektedir.

4. Nesneye Yönelik Tasarım Süreci

Bu bölümde, geliştirilen sistemin işleyişini görsel ve açıklama olarak ifade eden sözleşme ile tasarım, kullanım senaryoları, UML sınıf, sequence ve activity diyagramlarına yer verilmiştir. Diyagramlar, sistemin kullanıcılarla ve kendi içindeki bileşenlerle nasıl etkileşime girdiğini göstermeyi amaçlamaktadır. Böylece, sistemin mantıksal yapısı ve ayrıntıları daha iyi anlaşılabilir hale gelmiştir.

4.1. Sözleşme

Sözleşme No: 1 – Harcama Belgesi Oluşturma ve Muhasebeye Bildirim

İşlem: generate_expense_pdf(self, expense_data, employee_data, save_path=None)

Çapraz Başvurular: Harcama bildirimi senaryosu, muhasebe izleme senaryosu

Ön Koşullar:

- Çalışan sisteme giriş yapmış olmalıdır.
- Çalışanın daha önceden sisteme girilmiş ve yönetici tarafından onaylanmış en az bir harcaması olmalıdır.

Son Koşullar:

- Harcama bilgileri kullanılarak bir çalışan tarafından PDF belge oluşturulabilmektedir.
- Belge çalışan tarafından sistem dışına aktarılabilir (PDF olarak indirilir).
- Aynı harcama girdisi, muhasebe dashboardunda görünür hale gelmiştir.
- Muhasebe birimi sistem üzerinden harcamayı görüntüleyebilir, ancak belgenin kendisi fiziksel olarak elden teslim alınacaktır. Muhasebe birimi bu belgeyi sistem üzerinden oluşturamamaktadır.

4.2. Kullanım Senaryosu (Use-Case)

Bu bölümde, geliştirilecek olan bütçe ve harcama yönetim sistemine ait temel kullanım senaryoları tanımlanmıştır. Sistemde farklı roller üstlenen kullanıcıların (Üst Yönetici, Çalışan, Muhasebe, Yönetici) gerçekleştireceği işlemler, aktörler ve bu işlemlerin kapsamı detaylandırılmıştır. Diyagramda; üst yöneticinin bütçe tahmini yapması, birimler bazında bütçe tanımlaması ve raporlama faaliyetleri; yöneticinin biriminde talep edilen harcamaları onaylaması ya da reddetmesi; çalışanın harcama girişleri, durum takibi ve harcamaların belgelenmesi gibi işlemleri; muhasebe biriminin ise tazmin işlemlerini yürütmesi gibi temel işlevler gösterilmiştir. "Include" ve "Extend" gibi ilişkiler kullanılarak, senaryoların birbirleriyle olan bağımlılıkları ve alt faaliyetleri de görselleştirilmiştir.

Görsel 4.1. Use Case Diyagramı

1. Senaryo	Kurumsal Yönetim Sistemine Giriş
Birinci Aktör	Üst Yönetici/Yönetici/Çalışan/Muhasebe
İlgililer ve Beklentiler	Kullanıcıların sorunsuz bir şekilde sisteme giriş yapabilmelidir. Kullanıcı mailleri ve şifreleri şirket tarafından verilmiştir. Kurumsal Yönetim Sistemine kendileri kaydolamazlar.
Ön Koşullar	Kullanıcı giriş yapabilmek için bilgileri doğru girebilmelidir

Son Koşullar	Kullanıcı başarıyla giriş yaptıktan sonra rolüne ait arayüz ekranına yönlendirilir
Ana Akış	1. Kullanıcı bilgilerini girer 2. Kullanıcı bilgilerini girip giriş yaptıktan sonra her kullanıcı kendi kullanıcı rolüne özel olan uygulama arayüzüne yönlendirilir.
Alternatif Akış	1. Girilen bilgiler yanlışsa kullanıcı sisteme giriş yapamaz ve hata mesajı alır.

2. Senaryo	Her Birim ve Kalem için Bütçe Tanımlama
Birinci Aktör	Üst Yönetici
İlgililer ve Beklentiler	Üst Yönetici: Departman ve harcama kalemlerine ait bütçeleri ve eşik değerleri belirleyerek harcamaların kontrol altında tutulmasını ister. Yönetici: Kendi biriminin harcama sınırlarını bilerek personelin taleplerini buna göre değerlendirir. Muhasebe: Tazmin sürecinde belirlenen limitleri dikkate alarak tazmin işlemini gerçekleştirir.
Ön Koşullar	Üst Yönetici sisteme giriş yapmış olmalıdır.
Son Koşullar	Departmanlar ve harcama kalemleri sistemde tanımlı olmalıdır.
Ana Akış	1. Üst yönetici sisteme giriş yapar. 2. Sistem mevcut departman ve harcama kalemlerini listeler. 3. Üst yönetim her birim ve kalem için aylık bütçe değerini girer, girilen bilgileri düzenleyebilir veya silebilir. 4. Her birim kalem için eşik oranını belirler (%10 gibi). 5. Üst Yönetim, sistem üzerinden çalışanları ve yaptıkları harcamaları listeler. 6. Birim/Kalem/Kişi Bazında Raporlama Yapar 7. Geçmiş verileri dikkate alarak gelecek yıl için bütçe tahmininde bulunur.

Alternatif Akış	3.a. Silme işlemi sırasında tablodan silinmek istenen veri seçilmelidir, aksi takdirde hata verir. 3.b. Bütçe bilgileri içerisinde yer alan kişi bazlı harcama limit değeri bütçe değerinden fazla olmamalıdır. Aksi takdirde sistem uyarı verir ve eklenmek istenen/güncellenmek istenen bilgiler kaydedilmez

3. Senaryo	Harcama ekleme
Birincil Aktör	Çalışan
İlgililer ve Beklentileri	Çalışan: Seçtiği kaleme göre harcama girmek ister. Çalışan: Önceden eklemiş olduğu talepleri görüntüleyebilmek ister. Yönetici: Çalışanın girdiği harcamaları listeleyebilmek ister.
Ön Koşullar	Çalışan sisteme giriş yapmıştır. Çalışanın bağlı bulunduğu birime tanımlı harcama kalemi vardır.
Son Koşullar	Harcama talebi başarıyla sisteme eklenmelidir. Yeni talep kullanıcı ve yönetici ekranında görüntülenebilmelidir.
Ana Senaryo	1. Sistem çalışanın bağlı bulunduğu departmana tanımlı harcama kalemi ve bütçeleri gösterir. 2. Çalışan istediği kalemden birisini seçer ve alt bölümdeki harcama miktarı alanına istediği miktarı girer. 3. Çalışan “Talep Oluştur” butonuna basarak harcamasını oluşturur. 4. Çalışan “Geçmiş Harcamalar” bölümünden oluşturduğu harcamayı görüntüleyebilmelidir.

	5. Harcama yönetim tarafından onaylandıktan sonra çalışan Harcama Tazmin Belgesi'ni alarak muhasebeye gider.

4. Senaryo	Harcama Tazmin Belgesi Oluşturma
Birincil Aktör	Çalışan
İlgililer ve Beklentileri	Çalışan: Seçtiği harcamanın Harcama Tazmin Belgesi'ni görmek ve indirebilmek ister.
Ön Koşullar	Çalışan sisteme giriş yapmıştır. Çalışanın bağlı bulunduğu birime tanımlı harcama kalemi vardır. Çalışanın en az bir tane onaylanmış harcama talebi bulunmaktadır.
Son Koşullar	Çalışan seçtiği harcamaya ait Harcama Tazmin Belgesi'ni oluşturup indirebilmelidir.
Ana Senaryo	1. Çalışan sisteme giriş yapar ve "Geçmiş Harcamalar" bölümüne ilerler. 2. Çalışan "Onaylandi" durumundaki harcamalarından birini seçer. 3. Çalışan "Harcama Talebi Detayları" bölümünde yer alan Harcama Detay Raporu" yazan butona basarak belgeyi oluşturur. 4. Belge masaüstünde oluşturulur ve otomatik olarak açılır.
Alternatif Akış	2.a.Çalışan onaylanmamış bir harcama için belge almak ister. 2.a.1. "Harcama Detay Raporu" ön yüzde bloke olarak gösterilir ve kullanıcının eylemi engellenir.

5.Senaryo	Muhasebe İşlemleri
Birincil Aktör	Muhasebe
İlgililer ve Beklentileri	Çalışan: Harcamasını tazmin etmek ister. Muhasebe: Tazmin taleplerinin kalemini düzeltmek ve onaylamak ister.
Ön Koşullar	Muhasebe sisteme giriş yapmıştır. Çalışan Harcama Tazmin Belgesi'ni elden vermiştir. Çalışanın girdiği harcamaları yönetici onaylamıştır.
Son Koşullar	Harcama talebi tazmin edilmiştir.
Ana Senaryo	1. Sistem yönetici onayından geçmiş tazmin taleplerini gösterir. 2. Muhasebe istediği taleplerden birisini seçer ve ayrıntılarını görüntüler. 3. Muhasebe "Tazmin Et" butonuna basarak harcamayı tazmin eder.
Alternatif Senaryo	2a. Çalışan harcama kalemini yanlış girmiştir. 1. Muhasebe harcama kalemini düzeltir. 3a. Harcama kişinin harcayabileceği limiti aşmıştır. 1. Harcama limit ve eşik değeri kadar tazmin edilerek kısmi tazmin gerçekleşir. 3b. Harcama birimin kalem bütçesini aşmıştır. 1. Son kez harcama aşım miktarı ile kabul edilir. 2. Kullanıcılara bir daha tazmin talebinde bulunamayacakları bildirimi gönderilir. 3c. Tazmin talebi reddedilir. 3d. Muhasebe "Tüm Talepler" bölümünden tazmin edilmiş ya da reddedilmiş tüm talepleri görüntüleyebilir.

6.Senaryo	Harcama Talebini Onaylama
Birincil Aktör	Birim Yöneticisi
İlgililer ve Beklentileri	Yönetici: Harcamaları doğru ve zamanında kontrol edip karar vermek ister. Çalışan: Talebinin hızlı ve adil şekilde değerlendirildiğini görmek ister. Üst Yönetim: Harcamaların birim limitleri ve eşik değerler dahilinde yönetildiğinden emin olmak ister.
Ön Koşullar	Yönetici sisteme başarılı bir şekilde giriş yapmış olmalıdır. Yöneticiye ait birime ait bekleyen harcama talepleri mevcut olmalıdır.
Son Koşullar	Harcama durumu "Onaylandı" olarak güncellenir. Sistem tabloyu yeniler ve yöneticiye son durumu gösterir.
Ana Senaryo	1. Yönetici sisteme giriş yapar. 2. Sistem, yöneticinin birimine ait harcama taleplerini listeler. 3. Yönetici "Beklemede" durumundaki bir harcamayı inceler. 4. Yönetici harcamayı uygun bulur ve "Onayla" butonuna tıklar. 5. Sistem, veritabanında ilgili kaydın durumunu "Onaylandi" olarak günceller. 6. Sistem tabloyu günceller, özet kutucuklardaki sayılar da güncellenir.
Alternatif Senaryo	**3a.** Yönetici harcamayı yetersiz belge veya açıklama nedeniyle uygun bulmaz. 1. "Reddet" butonuna tıklar. 2. Sistem durumu "Reddedildi" olarak günceller. 3. Tablo ve istatistik kutucukları güncellenir.

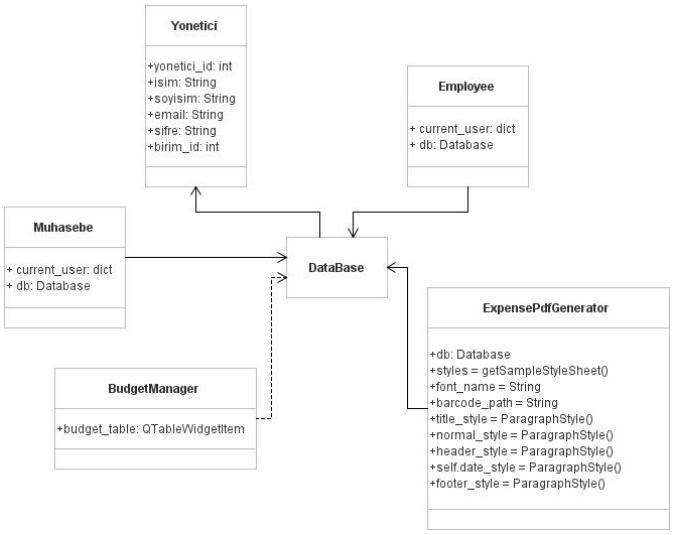
	4a. Onaylanmak istenen harcama kalemi, limit ve eşik değer sınırını aşmaktadır. 1. Yönetici buna rağmen onay verebilir veya reddedebilir. 2. Sistem karar sonucuna göre güncelleme yapar.

4.3. Problem Uzayının Modellenmesi

Sınıf Diyagramı

Bu bölümde, geliştirilen gider yönetim sisteminin problem uzayı modellenmiştir. Problem uzayı, sistemin çözmeyi hedeflediği temel ihtiyaçları ve iş süreçlerini tanımlamak amacıyla kullanılır. Bu bağlamda, sistemde yer alacak aktörlerin (Üst Yönetici, Yönetici, Çalışan, Muhasebe) gerçekleştireceği temel işlemler ve bu işlemlerin birbirleriyle olan ilişkileri UML Sınıf Diyagramı aracılığıyla görselleştirilmiştir.

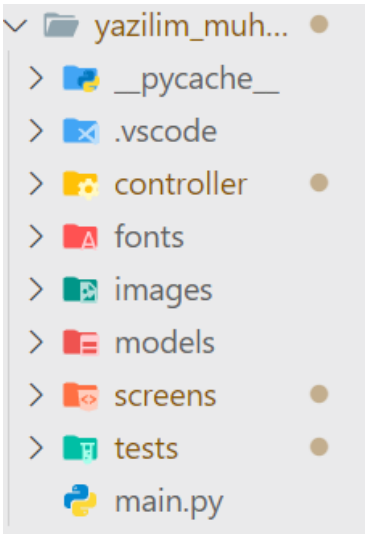
Görsel 4.3.1 - Gider Yönetim Sistemi UML Sınıf Diyagramı



Görsel 4.3.2 - Class Diyagram

4.4. Tasarım Kalıbı (MVC Mimarisi)

Bu projede Model-View-Controller (MVC) yazılım mimarisi benimsenmiştir. Bu mimari, kodun görevlerine göre ayrılmasını ve daha modüler, okunabilir, sürdürülebilir bir yapı oluşturmayı sağlamaktadır. Görsel 4.4'te sistem gerçeklemesinde oluşturulan yapı gösterilmiştir.

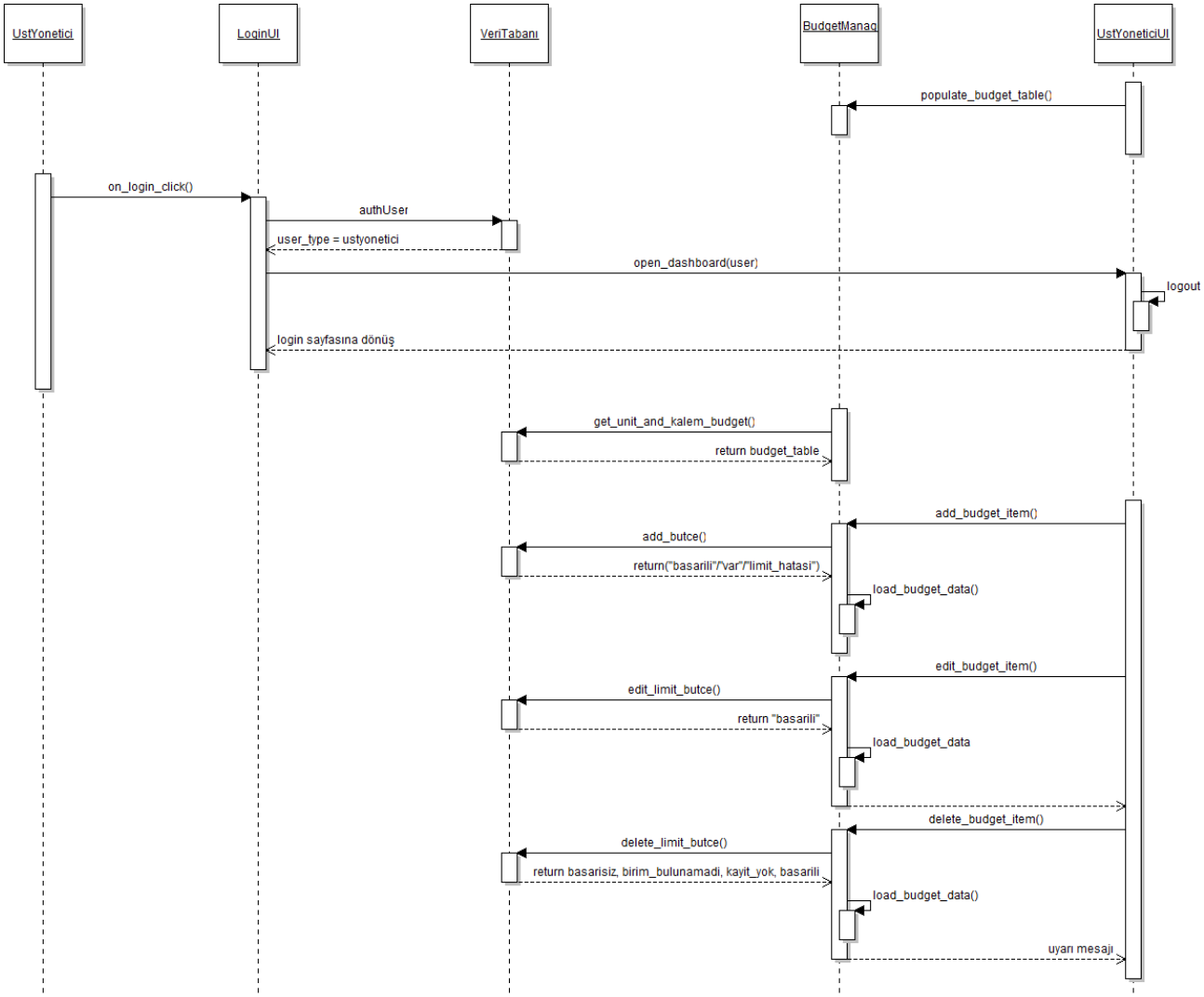


Görsel 4.4 - MVC Tasarım Kalıbı

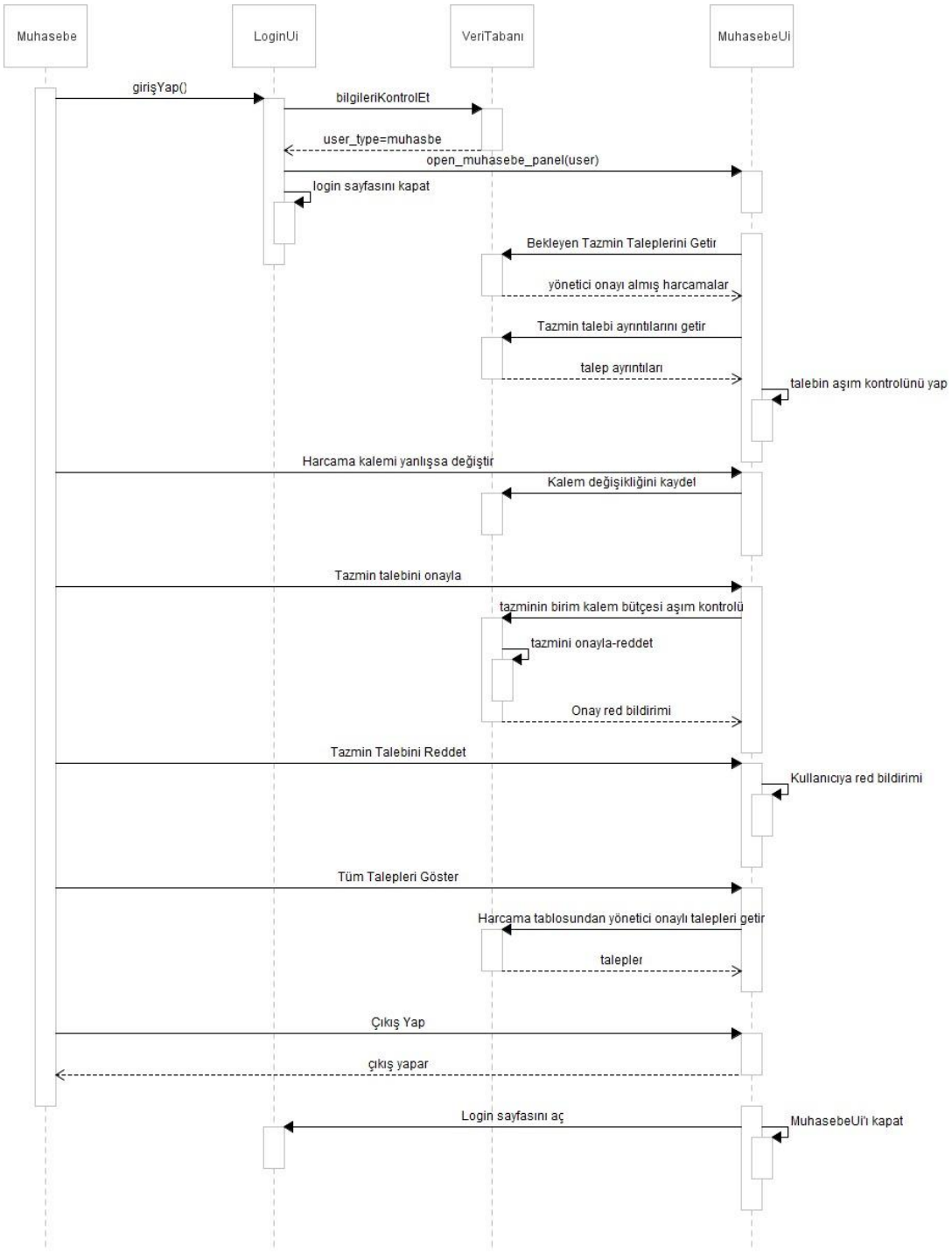
Bu yapıda screens klasörü MVC mimarisindeki views yapısını temsil etmektedir. Arayüze ait gerekli dosyalar burada bulunmaktadır. Controller içerisinde bulunan dosyalarla arayüz ile veritabanı arasındaki bağlantıyı sağlayacak metodlar bulunmaktadır. Veritabanına ait yapılar models klasörü içerisinde yer almaktadır.

4.5. Sequence Diagramları

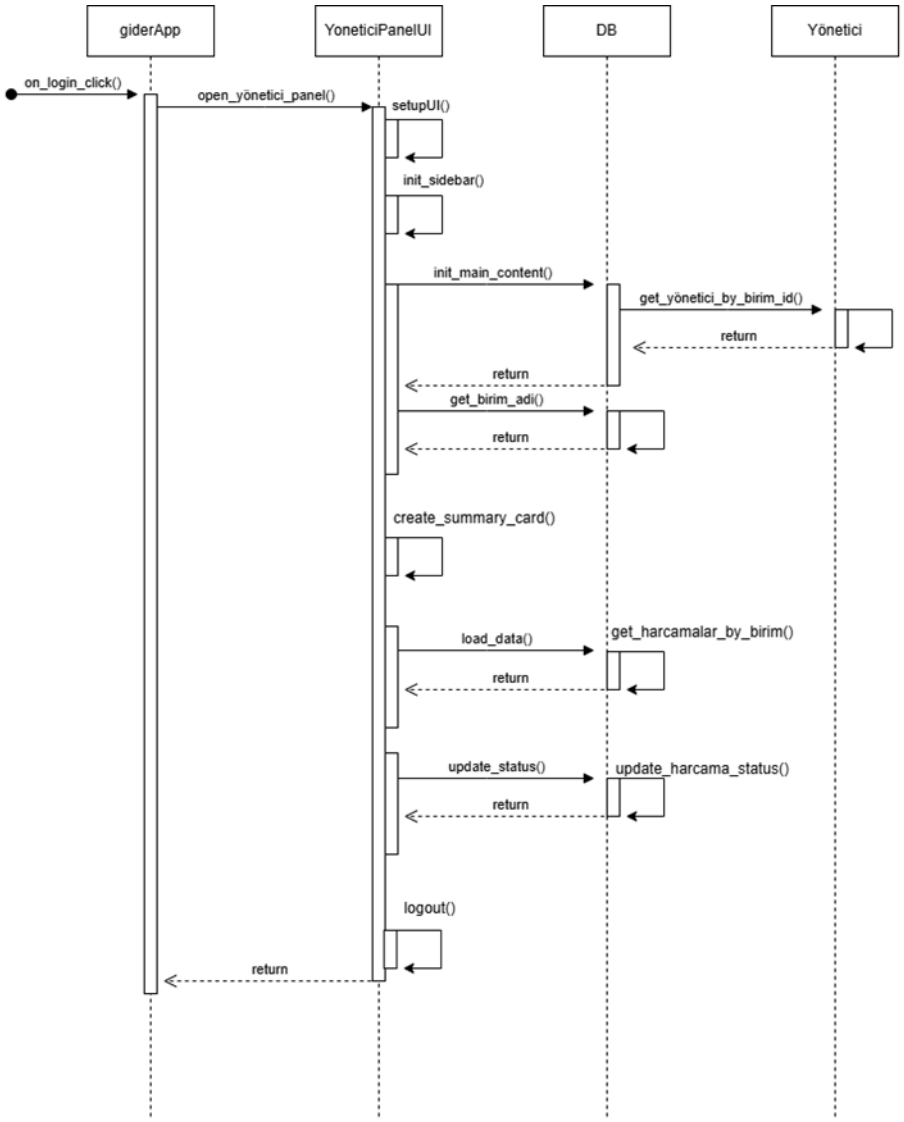
4.5.1. Üst Yönetici:



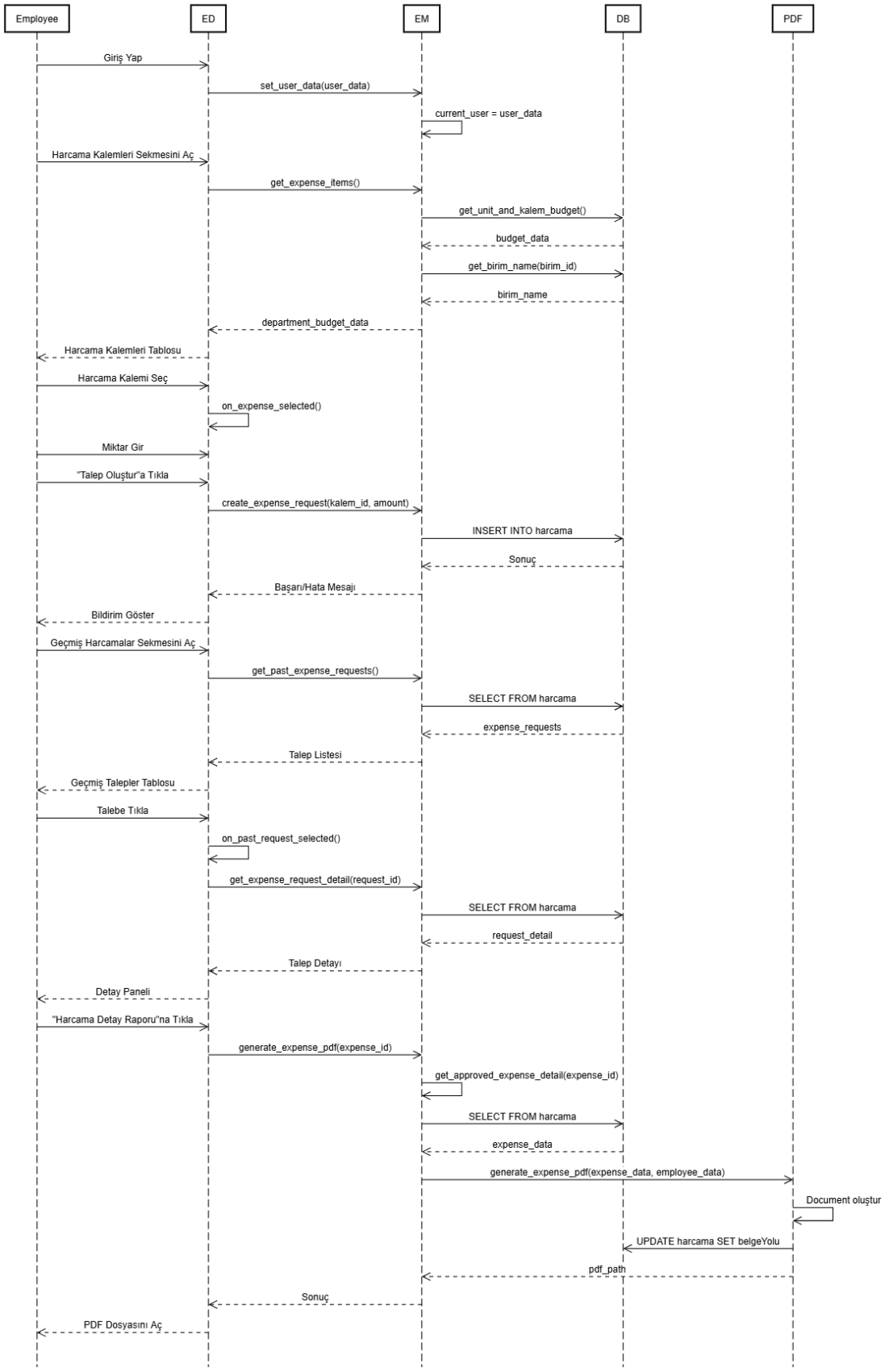
4.5.2. Muhasebe:



4.5.3. Yönetici:

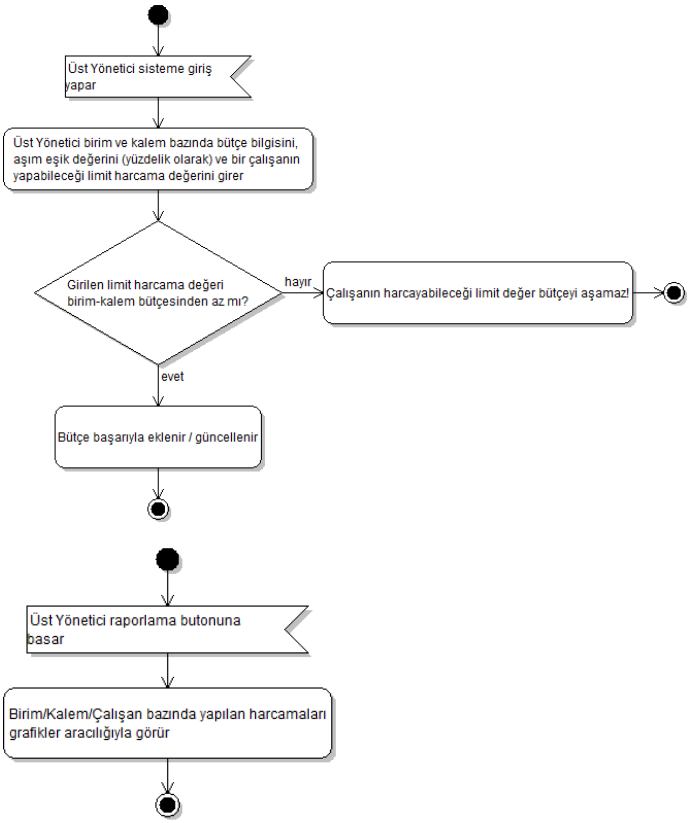


4.5.4. Çalışan:

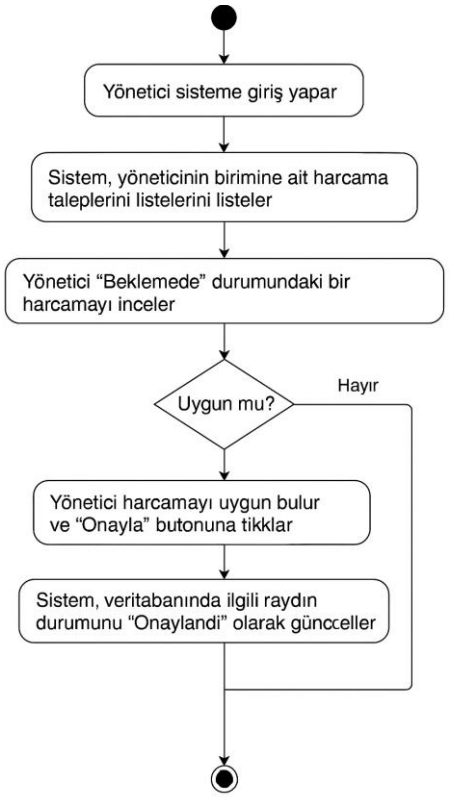

Employee
ED
EM
DB
PDF
Giriş Yap
set_user_data(user_data)
current_user = user_data
Harcama Kalemleri Sekmesini Aç
get_expense_items()
get_unit_and_kalem_budget()
budget_data
get_birim_name(birim_id)
birim_name
department_budget_data
Harcama Kalemleri Tablosu
Harcama Kalemi Seç
on_expense_selected()
Miktar Gir
"Talep Oluştur"a Tıkla
create_expense_request(kalem_id, amount)
INSERT INTO harcama
Sonuç
Başarı/Hata Mesajı
Bildirim Göster
Geçmiş Harcamalar Sekmesini Aç
get_past_expense_requests()
SELECT FROM harcama
expense_requests
Talep Listesi
Geçmiş Talepler Tablosu
Talebe Tıkla
on_past_request_selected()
get_expense_request_detail(request_id)
SELECT FROM harcama
request_detail
Talep Detayı
Detay Paneli
"Harcama Detay Raporu"na Tıkla
generate_expense_pdf(expense_id)
get_approved_expense_detail(expense_id)
SELECT FROM harcama
expense_data
generate_expense_pdf(expense_data, employee_data)
Document oluştur
UPDATE harcama SET belgeYolu
pdf_path
Sonuç
PDF Dosyasını Aç

4.6 Activity Diyagramları

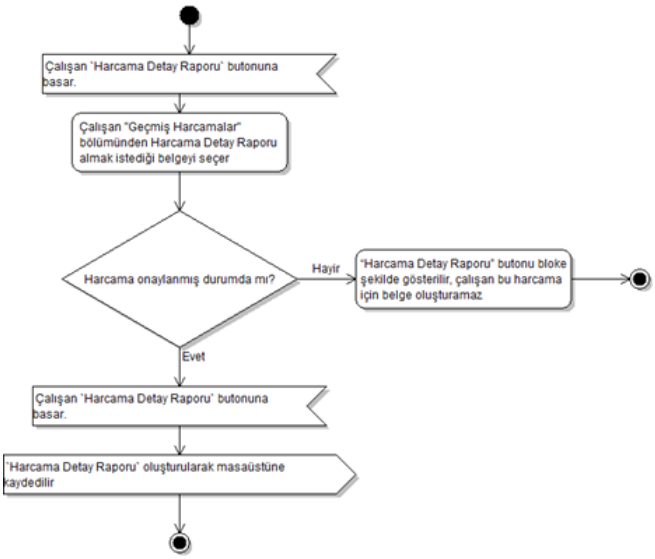
4.6.1 Üst Yönetici



4.6.2. Yönetici

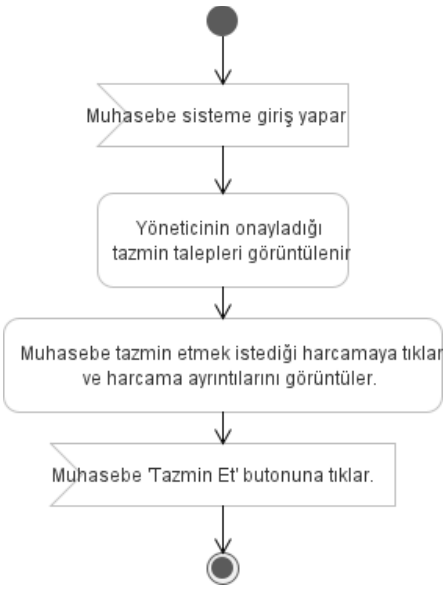


4.6.3. Çalışan

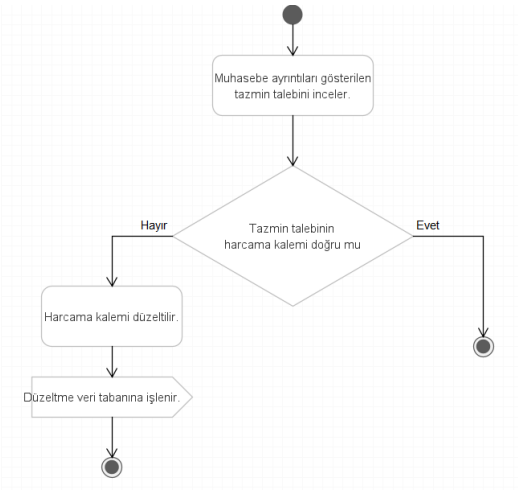


4.6.4. Muhasebe

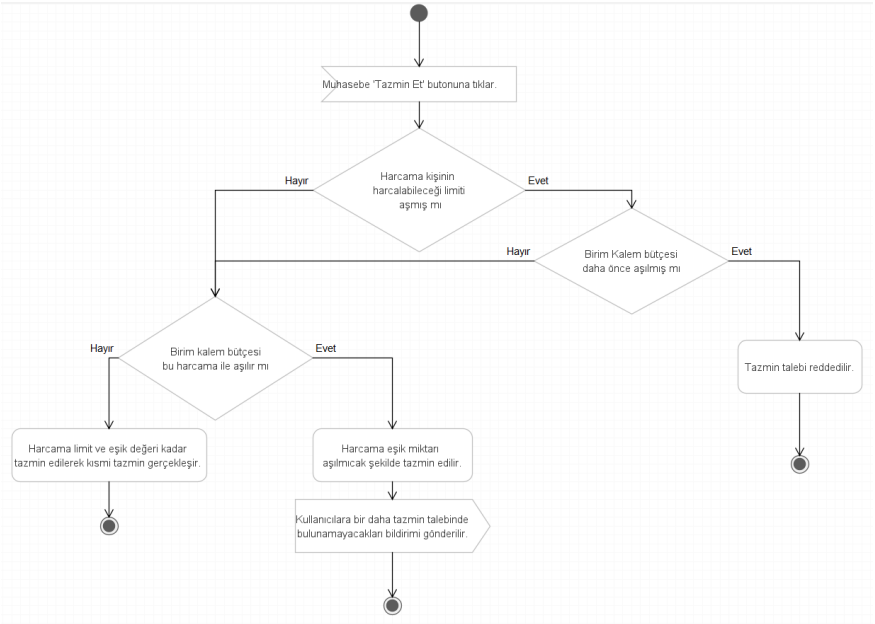
a. Ana Senaryo



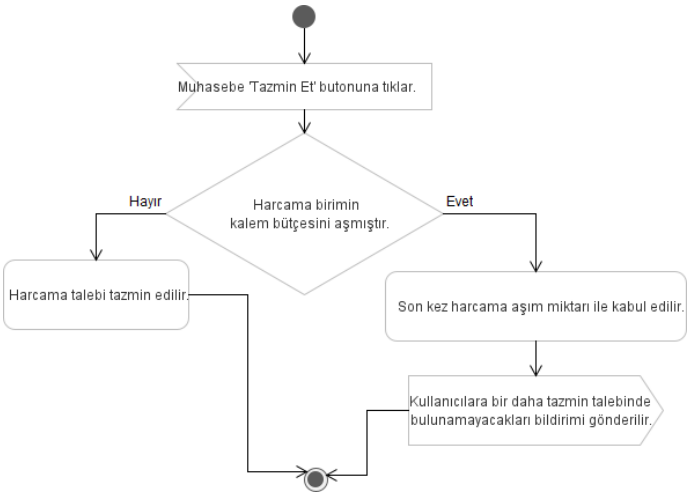
Alternatif Senaryo 2a:



b. Alternatif Senaryo 3a:



c. Alternatif Senaryo 3b:

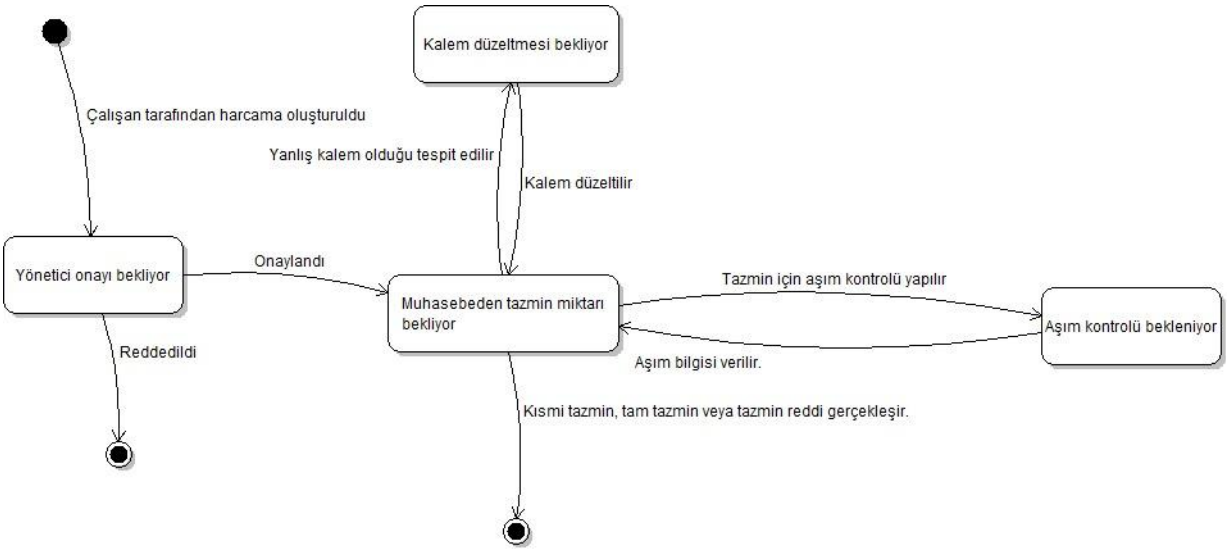


d. Alternatif Senaryo 3c:

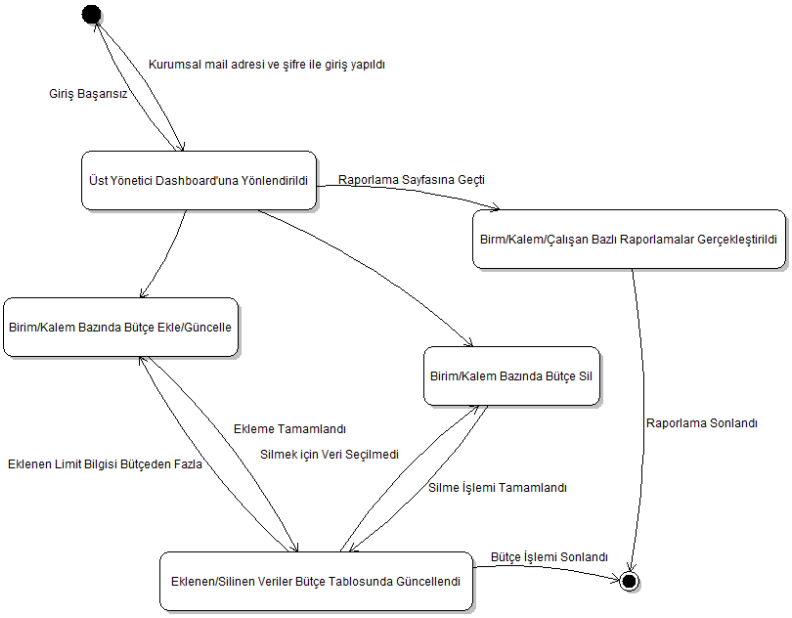
e. Alternatif Senaryo 3d:

4.7 State (Durum) Diyagramları

Sistem State (Durum) Diyagramı



Üst Yönetici State (Durum) Diyagramı



5. Birim Test Sınamaları

5.1. Üst Yönetici Birim Testleri

Bu birimdeki testler Ebru Kılıç tarafından yazılmıştır. Burada üst yönetici tarafından oluşturulan bütçe bilgisinin veri tabanına doğru bir şekilde eklenip eklenmediği, eğer zaten belirtilen birim ve kaleme ait bütçe bilgisi varsa bu konuda ekleyen kişiye uyarı verilip verilmediği test edilmektedir. Ekleme işlemine ek olarak bütçe bilgisinin düzenlenmesi hususunda da gerekli testler yapılmıştır.

```
class TestAddBudget(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.app = QApplication(sys.argv)

    def setUp(self):
        self.table_widget = QTableWidget()
        self.budget_manager = BudgetManager(self.table_widget)
        self.budget_manager.budget_table = mock.Mock(spec=QtWidgets.QTableWidget)
        self.table = QTableWidget()
        self.table.setColumnCount(7)

        headers = ["Birim Adı", "Kalem Adı", "Toplam Bütçe", "Kullanılan Bütçe",
                   "Kalan Bütçe", "Limit Bütçe", "Aşım Oranı"]
        self.table.setHorizontalHeaderLabels(headers)

        self.budget_manager = BudgetManager(self.table)

        self.db_patcher = patch('models.database.Database')
        self.mock_db = self.db_patcher.start()

    def tearDown(self):
        self.db_patcher.stop()
```

```
@patch('PyQt5.QtWidgets.QDialog.exec_')
@patch('PyQt5.QtWidgets.QMessageBox.warning')
@patch('PyQt5.QtWidgets.QMessageBox.information')
def test_add_budget_item_success(self, mock_info, mock_warning, mock_exec):
    instance = self.mock_db.return_value
    instance.add_butce.return_value = "basarili"

    instance.cursor = MagicMock()
    instance.cursor.fetchall.return_value = [("Taksi",), ("Benzin",)]

    mock_exec.return_value = 1

    with patch('PyQt5.QtWidgets.QComboBox.currentData', side_effect=["Satış", "Benzin"]), \
        patch('PyQt5.QtWidgets.QLineEdit.text', side_effect=["5000", "10000"]), \
        patch('PyQt5.QtWidgets.QSlider.value', return_value=20), \
        patch.object(self.budget_manager, 'load_budget_data'):

        self.budget_manager.add_budget_item()

        instance.add_butce.assert_called_once_with("Benzin", "Satış", 10000.0, 5000.0, 20)

        mock_info.assert_called_once()

        self.budget_manager.load_budget_data.assert_called_once()
```

```
ModuleNotFoundError: No module named 'yazilim_muh_proje_myvers'
PS C:\Users> python -u "c:\Users\Lenovo\Desktop\yazilim-muh-proje-yeni\yazilim_muh_proje_myvers\tests\test_add_budget.py"
...
----------------------------------------------------------------------
Ran 3 tests in 0.620s

OK
```

```python
@patch('PyQt5.QtWidgets.QDialog.exec_')
@patch('PyQt5.QtWidgets.QMessageBox.warning')
def test_add_budget_item_already_exists(self, mock_warning, mock_exec):
    instance = self.mock_db.return_value
    instance.add_butce.return_value = "var"

    instance.cursor = MagicMock()
    instance.cursor.fetchall.return_value = [("Taksi",), ("Benzin",)]

    mock_exec.return_value = 1  # QDialog.Accepted

    with patch('PyQt5.QtWidgets.QComboBox.currentData', side_effect=["Satış", "Benzin"]), \
        patch('PyQt5.QtWidgets.QLineEdit.text', side_effect=["5000", "10000"]), \
        patch('PyQt5.QtWidgets.QSlider.value', return_value=20), \
        patch.object(self.budget_manager, 'load_budget_data'):

        self.budget_manager.add_budget_item()

        mock_warning.assert_called_once()

        self.budget_manager.load_budget_data.assert_not_called()
```

```python
@patch('models.database.sqlite3.connect')
def test_edit_limit_butce_success(self, mock_connect):
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_connect.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    # kalem_id bulundu
    mock_cursor.fetchone.side_effect = [
        (5,),           # SELECT kalemId sonucu
        ('some_data',)  # SELECT * FROM birim_kalem_butcesi sonucu
    ]

    manager = Database("gider_new.db")
    result = manager.edit_limit_butce("Benzin", 1, 10000, 8000, 10)

    self.assertEqual(result, "basarili")
    self.assertTrue(mock_conn.commit.called)
    self.assertTrue(mock_conn.close.called)

@patch('models.database.sqlite3.connect')
def test_edit_limit_butce_no_kalem(self, mock_connect):
    mock_conn = MagicMock()
    mock_cursor = MagicMock()
    mock_connect.return_value = mock_conn
    mock_conn.cursor.return_value = mock_cursor

    mock_cursor.fetchone.return_value = None  #* kalem bulunamadı -> bu durumda basarisiz olarak buluyor mu test edildi

    manager = Database("gider_new.db")
    result = manager.edit_limit_butce("YanlışKalem", 1, 10000, 8000, 10)

    self.assertEqual(result, "basarisiz")
```

```
PS C:\Users\Lenovo\Desktop\yazilim-muh-proje-yeni> python -u "c:\Users\Lenovo\Desktop\yazilim-muh-proje-yeni\yazilim_muh_proje_myvers\tests\test_edit_budget.py"
..
----------------------------------------------------------------------
Ran 2 tests in 0.534s

OK
```

5.2. Yönetici Birim Testleri

Bu birimdeki testler Meryem İbrahimoğlu tarafından yazılmıştır. Burada arayüz ve veritabanı arasında gerçekleşen işlemlerin doğru çalışıp çalışmadığı test edilmiştir.

```
Mariam-Eldesouky, 7 days ago | 1 author (Mariam-Eldesouky)
class TestYoneticiPanel(unittest.TestCase):
    def setUp(self):
        self.panel = YoneticiPanelUI()
        self.birim_id = 1  # Make sure this exists in your test DB
        self.panel.setupUi(self.panel, self.birim_id)

    def test_load_data_row_count(self):
        """Test if the table loads data rows for the given birim."""
        self.panel.load_data()
        row_count = self.panel.table.rowCount()
        self.assertGreater(row_count, 0, "Table should have at least one row")

    def test_status_update_logic(self):
        """Test if the status update mechanism works logically (no DB assertion)."""
        old_method = self.panel.db.update_harcama_status
        self.called = False

        def mock_update(harcama_id, new_status):
            self.called = True
            self.assertIn(new_status, ["Onaylandi", "Reddedildi"])

        self.panel.db.update_harcama_status = mock_update
        self.panel.update_status(1, "Onaylandi")
        self.assertTrue(self.called)

        # Restore original method
        self.panel.db.update_harcama_status = old_method

    def test_yonetici_object_fetch(self):
        """Test if the Yonetici object is returned correctly from DB."""
        db = Database()
        yonetici = db.get_yonetici_by_birim_id(self.birim_id)
        self.assertIsInstance(yonetici, Yonetici)
        self.assertEqual(yonetici.birim_id, self.birim_id)
        self.assertTrue(yonetici.get_full_name())  # Check string output
```

```
> python -u "c:\Users\Lenovo\Desktop\yazilim-muh-proje
-yeni\yazilim_muh_proje_myvers\tests\test_yönetici_panel.py"
....
----------------------------------------------------------------------
Ran 4 tests in 8.501s

OK
```

```
Mariam-Eldesouky, 7 days ago | 1 author (Mariam-Eldesouky)
class TestYoneticiPanelAdvanced(unittest.TestCase):
    def setUp(self):
        self.panel = YoneticiPanelUI()
        self.birim_id = 1  # Ensure this birimId exists in test DB
        self.panel.setupUi(self.panel, self.birim_id)

    def test_column_headers(self):
        """Check if all expected column headers are set correctly."""
        self.panel.load_data()  # Ensure headers are set

        expected_headers = [
            "ID", "Ad Soyad", "Kalem", "Tutar", "Tazmin", "Açıklama",
            "Durum", "Tarih", "Limit Aşıldı mı", "Şuan Aşım Var mı", "Aşım Miktarı", "İşlem"
        ]

        self.assertEqual(self.panel.table.columnCount(), len(expected_headers), "Column count mismatch")

        for i, expected in enumerate(expected_headers):
            header_item = self.panel.table.horizontalHeaderItem(i)
            self.assertIsNotNone(header_item, f"Header at index {i} is None")
            self.assertEqual(header_item.text(), expected)


    def test_summary_card_value_labels_are_strings(self):
        """Ensure each summary card value is a string (e.g., '0', '5', etc)."""
        self.panel.load_data()

        cards = [self.panel.card_beklemede, self.panel.card_onaylandi, self.panel.card_reddedildi]
        for card in cards:
            # Access the QLabel assigned in `create_summary_card`
            value_label = card.value_label  # Access via dynamic property
            self.assertIsInstance(value_label.text(), str, "Card value is not a string")

if __name__ == "__main__":
    unittest.main()
```

```
                                                    > python -u "c:\Users\Lenovo\Desktop\yazilim-muh-proje
-yeni\yazilim_muh_proje_myvers\tests\test_yönetici_panel2.py"
..
----------------------------------------------------------------------
Ran 2 tests in 0.347s

OK
```

5.3. Muhasebe Birim Testleri

Bu birimdeki testler İclal Ertürk tarafından yazılmıştır. Testlerde, bütçe aşımı durumunda harcamanın tazmin edilmediği, bütçenin ilk defa aşıldığı durumda kullanıcılara bildirim gönderildiği, yönetici onayı almış harcamaların çekilmesi ve bu süreçlerde ilgili veri tabanı işlemlerinin doğru şekilde gerçekleşip gerçekleşmediği test edilmiştir.

```python
def test_get_approved_expenses(self):
    mock_db = MagicMock()
    # Geri döndürülecek örnek veri
    expected_result = [
        (1, "email@example.com", "Birim A", "Kalem X", 100.0, "2025-05-17", "Onaylandi"),
        (2, "email2@example.com", "Birim B", "Kalem Y", 200.0, "2025-05-16", "Onaylandi")
    ]
    mock_db.cursor.fetchall.return_value = expected_result

    muhasebe = Muhasebe.__new__(Muhasebe)  # Yapıcıyı çağırmadan örnek oluştur
    muhasebe.db = mock_db

    result = muhasebe.get_approved_expenses()

    # Sorgu doğru çalıştı mı kontrolü
    mock_db.cursor.execute.assert_called_once()
    self.assertEqual(result, expected_result)
```

```python
@patch("controller.muhasebe_class.QtWidgets.QMessageBox")
def test_on_approve_button_budget_already_exceeded(self, mock_messagebox):
    mock_table = MagicMock()
    mock_table.currentRow.return_value = 0
    mock_table.item.return_value.text.return_value = "1"  # harcama_id

    mock_radio = MagicMock()
    mock_radio.isChecked.return_value = True  # Full reimbursement seçili

    mock_detail_amount = MagicMock()
    mock_detail_amount.text.return_value = "150.0 ₺"  # tazmin miktarı

    mock_parent = MagicMock()
    mock_parent.detail_amount = mock_detail_amount

    mock_reimburse_amount = MagicMock()

    load_func = MagicMock()

    mock_db = MagicMock()
    mock_db.cursor.fetchone.side_effect = [
        (10, 20),      # birimId, kalemId
        (100.0,),      # limitButce
        (1,),          # butceAsildi = 1
        (90.0,)        # toplam_harcanan
    ]
    mock_db.cursor.fetchall.return_value = [(99,)]  # kullanıcı listesi

    with patch("controller.muhasebe_class.Database", return_value=mock_db):
        muhasebe = Muhasebe(mock_table, mock_radio, mock_reimburse_amount, load_func, user_data=None, parent=mock_parent)
        muhasebe.on_approve_button_clicked()

    mock_messagebox.warning.assert_called_with(None, "Bütçe Aşıldı", "Bütçe zaten aşıldı. Bu harcama onaylanamaz.")

    mock_db.cursor.execute.assert_any_call(
        "UPDATE harcama SET onayDurumu = 'Reddedildi', tazminDurumu = 'Reddedildi' WHERE harcamaId = ?", (1,)
    )
    mock_db.conn.commit.assert_called()
    load_func.assert_called()
```

```python
@patch('controller.muhasebe_class.Database')
def test_on_approve_button_budget_exceeded_first_time(self, MockDatabase):
    mock_db = MockDatabase.return_value
    mock_db.conn = MagicMock()
    mock_cursor = mock_db.cursor
    mock_cursor.fetchone.side_effect = [
        (1, 2),        # harcama: birimId, kalemId
        (100.0,),      # limitButce
        (0,),          # butceAsildi
        (90.0,),       # toplam harcanan
        ('Kalem A',),  # kalem adı
        (5, 100.0)     # birim_kalem_harcanan_butce: id, mevcut_butce
    ]
    mock_cursor.fetchall.return_value = [(10,), (11,)]  # kullanıcılar
    mock_table = MagicMock()
    mock_table.currentRow.return_value = 0
    mock_table.item.return_value.text.return_value = "1"  # harcamaId
    mock_radio = MagicMock()
    mock_radio.isChecked.return_value = False
    mock_spinbox = MagicMock()
    mock_spinbox.value.return_value = 20.0  # yeni talep edilen miktar
    mock_parent = MagicMock()
    mock_parent.detail_amount.text.return_value = "₺20"
    with patch.object(QtWidgets.QMessageBox, 'warning') as mock_warning:
        muhasebe = Muhasebe(
            pending_requests_table=mock_table,
            reimburse_full_radio=mock_radio,
            reimburse_amount=mock_spinbox,
            load_approved_expenses_to_table=lambda: None,
            user_data=None,
            parent=mock_parent
        )
        muhasebe.on_approve_button_clicked()

    mock_cursor.execute.assert_any_call(
        "UPDATE birim SET butceAsildi = 1 WHERE birimId = ?", (1,)
    )
    bildirim_calls = [
        c for c in mock_cursor.execute.call_args_list
        if "INSERT INTO bildirim" in c.args[0]
    ]
    assert len(bildirim_calls) == 2
    mock_cursor.execute.assert_any_call(
        "UPDATE harcama SET onayDurumu = 'Onaylandi', tazminDurumu = 'Onaylandi', tazminTutari = ? WHERE harcamaId = ?",
        (20.0, 1)
    )
    expected_new_amount = 100.0 + 20.0
    mock_cursor.execute.assert_any_call(
        "UPDATE birim_kalem_harcanan_butce SET harcanan_butce = ? WHERE id = ?",
        (expected_new_amount, 5)
    )
    mock_db.conn.commit.assert_called_once()
    mock_warning.assert_called_once()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\iclal\Documents\GitHub\yazilim-muh-proje\yazilim_muh_proje_myvers> & C:/Users/iclal/AppData/Local/Programs/Python/Python310/python.exe c:/Users/iclal
/Documents/GitHub/yazilim-muh-proje/yazilim_muh_proje_myvers/tests/test_muhasebe.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.013s

OK
PS C:\Users\iclal\Documents\GitHub\yazilim-muh-proje\yazilim_muh_proje_myvers>
```

5.4. Çalışan Birim Testleri

Bu birimdeki testler Sinem Sarak tarafından yazılmıştır. Burada kullanıcı bilgilerinin doğruluğu ve geçmiş harcama taleplerinin yükleme metodunun doğru çalışıp çalışmadığı test edilmektedir.

Aşağıda test metotları ve test çalıştırılması sonucu alınan çıktılara dair ekran görüntüleri yer almaktadır.

```python
class TestEmployeeDashboard(unittest.TestCase):
    def setUp(self):
        self.dashboard = EmployeeDashboard()
        self.employee1 = Employee(1, "Alice Smith", "Manager")
        self.employee2 = Employee(2, "Bob Johnson", "Engineer")

    def test_add_employee(self):
        self.assertTrue(self.dashboard.add_employee(self.employee1))
        self.assertEqual(self.dashboard.get_employee_count(), 1)
        self.assertIn(self.employee1, self.dashboard.employees)

    def test_add_invalid_employee_type(self):
        self.assertFalse(self.dashboard.add_employee("not an employee"))
        self.assertEqual(self.dashboard.get_employee_count(), 0)

    def test_get_employee_count(self):
        self.assertEqual(self.dashboard.get_employee_count(), 0)
        self.dashboard.add_employee(self.employee1)
        self.assertEqual(self.dashboard.get_employee_count(), 1)
        self.dashboard.add_employee(self.employee2)
        self.assertEqual(self.dashboard.get_employee_count(), 2)

    def test_get_employee_by_id(self):
        self.dashboard.add_employee(self.employee1)
        self.dashboard.add_employee(self.employee2)

        found_employee = self.dashboard.get_employee_by_id(1)
        self.assertIsNotNone(found_employee)
        self.assertEqual(found_employee.name, "Alice Smith")

        found_employee_2 = self.dashboard.get_employee_by_id(2)
        self.assertIsNotNone(found_employee_2)
        self.assertEqual(found_employee_2.name, "Bob Johnson")

        non_existent_employee = self.dashboard.get_employee_by_id(3)
        self.assertIsNone(non_existent_employee)
```

```python
class EmployeeDashboard:
    def __init__(self):
        self.employees = []

    def add_employee(self, employee):
        if isinstance(employee, Employee):
            self.employees.append(employee)
            return True      You, 5 minutes ago •
        return False

    def get_employee_count(self):
        return len(self.employees)

    def get_employee_by_id(self, employee_id):
        for emp in self.employees:
            if emp.employee_id == employee_id:
                return emp
        return None
```

```
PS C:\Users\sinem\Desktop\yazmühproje\yazilim-muh-proje\yazilim_muh_proje_myvers> & C:/Users/sinem/AppData/Local/Programs/Python/Python311/python.exe c:/Users/sinem/Desktop/yazmühproje/yazilim-muh-proje/yazilim_muh_proje_myvers/tests/test_employee_dashboard.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
```