

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
ELEKTRİK-ELEKTRONİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ



BLM-3722
YAZILIM MÜHENDİSLİĞİ
DİL KURSU OTOMASYON YAZILIMI

Dersin Sorumlu Öğretim Üyesi: Dr.Öğr.Üyesi Yunus Emre SELÇUK

21011111-Seden RAZİ
21501027-Batuhan HANGÜN
21501035-Önder GÖRMEZ
21501045-Büşra BAŞAER
21501081-Enes Doğan ŞANLI

ÖZET

Yazılım Mühendisliği dersi kapsamında hazırlanan bu projede, ‘Dil Kursu Otomasyon Sistemi’ nin geliştirilmesi için gerekli süreçler belirlenmiş ve süreç adımları belirli bir plana sadık olarak uygulanmıştır. Yazılım Mühendisliği dersi kapsamında öğrenildiği üzere projenin gereklilikleri tespit edilmiş, olası riskler belirlenmiş ve bu risklerin önlemleri alınmış, projenin kullanım şeması ve senaryoları çıkarılarak diyagramlarının çizilmiştir. Ardından proje planına uygun olarak kodlama yapılmış ve son olarak yazılan kod birim testlere tabi tutularak müşteriye teslimine uygun hale getirilmiştir.

İçindekiler

ÖZET	2
1. PROJE PLANI	4
1.1. Proje Alan Tanımı	4
1.2. Kabul ve Kısıtlar	4
1.3. Proje İş-Zaman Çizelgesi	5
1.4. Ekip Organizasyon Şeması	5
1.5. Risk Analiz Tablosu.....	5
2. İSTEKLERİN MODELLENMESİ	7
2.1. Kullanım Şeması (Use-Case).....	7
2.2. Kullanım Senaryoları.....	8
2.2.1. Şube ve Derslik Bilgisi Girilmesi	8
2.2.2. Yeni Ders Açılması.....	9
2.2.3. Öğrencinin Ders Kaydı.....	10
3. NESNEYE DAYALI MODELLEME.....	12
3.1. UML Sınıf Diyagramı	12
4. NESNEYE YÖNELİK TASARIM DİYAGRAMLARI	17
4.1. Etkinlik (Activity) Diyagramı.....	17
4.2. Durum (State) Diyagramı	18
4.3. Ardışıl (Sequence) Diyagram	19
5. İZLENEBİLİRLİK TABLOSU	20
6. BİRİM TESTLER.....	21
KAYNAKÇA.....	24

1. PROJE PLANI

1.1.Proje Alan Tanımı

Proje Adı: Dil Kursu Otomasyon Yazılımı

Proje Tanımı

“Bir Lisan Bir İnsan” dil kursu şirketinin öğretmen kadrosu genişletildiğinde veya yeni şubeler açıldığında takip ve güncellemelerinin dinamik yapılabilmesi için bir otomasyon yazılımı geliştirilmiştir.

Müşteri isteği doğrultusunda hazırlanan proje kapsamında çalışanların rollerine bağlı olmak kaydıyla şube ve derslik bilgileri kaydedilebilir veya güncellenebilir istenilmesi durumunda yeni şube açılabilir ve yeni öğrenci kaydı alınabilmektedir.

Proje Modülleri

-Öğrencinin Ders Kaydı :Bu modül ile kayıt koşullarını sağlayan öğrencinin ,kontenjanı bulunan derse kaydı yapılabilmektedir.

Yeni Ders Açılması:Bu modül ile açılması için yeterli talep gelen dersin uygun öğretmen ve derslikler belirlenerek açılması sağlanmaktadır.

Şube ve Derslik Bilgisi Girilmesi:Bu modül ile şube ve derslik bilgilerinin kaydedilmesi ve istenilmesi durumunda güncellenebilmesi sağlanmaktadır.

Proje Üyeleri:

- Batuhan HANGÜN
- Önder GÖRMEZ
- Enes Doğan ŞANLI
- Büşra BAŞAER
- Seden RAZİ

1.2.Kabul ve Kısıtlar

-Sisteme şube veya derslik ekleme işlemini yalnızca Sistem Yöneticisi yapabilir.

-Ders Kaydını yalnızca Kayıt Görevlisi yapabilir. Öğrenciler kaydolmak istedikleri ders için Kayıt Görevlisine gitmelidirler.

-Öğrencinin istediği derse kaydolmak için dersin açıldığı şubeye başvurmalıdır çünkü ders kaydı yalnızca dersin açıldığı şubeden yapılabilmektedir.

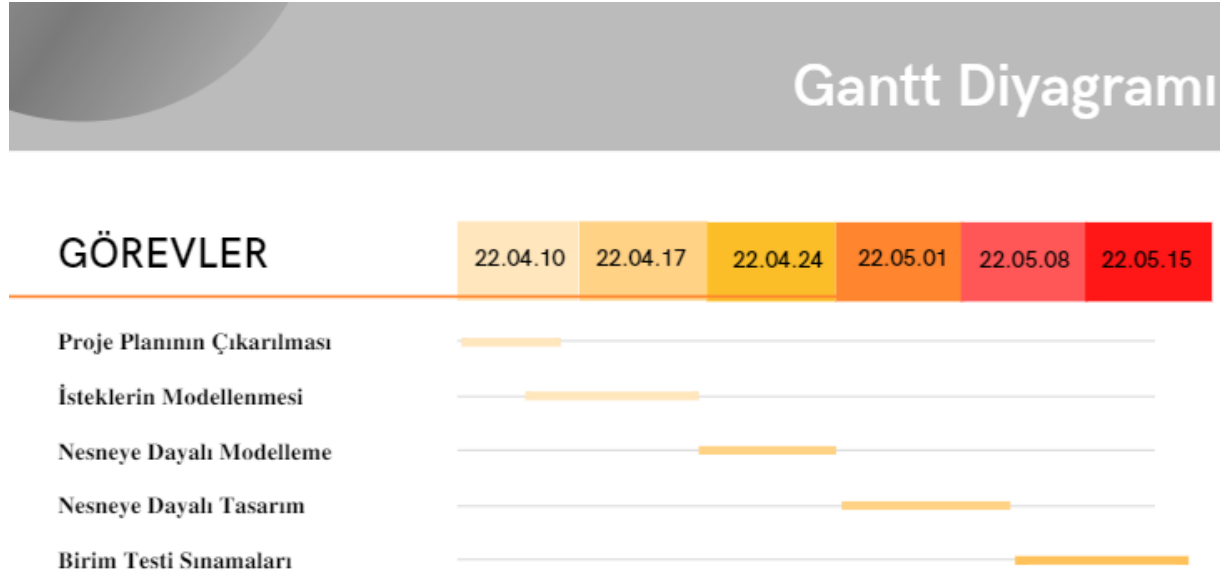
-Bir ders bir şubede tek bir grupta açılabilir.

-Pandemi vb. sebeplerden dersliklerin kullanım kapasitesinin tamamının kullanılmaması durumlarına karşılık ders kapasitesi ve dersliğin maksimum kapasitesi ayrı ayrı tutulmaktadır.

-Öğrencinin ders seçebilmesi için öncelikle kurumda kaydının bulunması gerekmektedir.

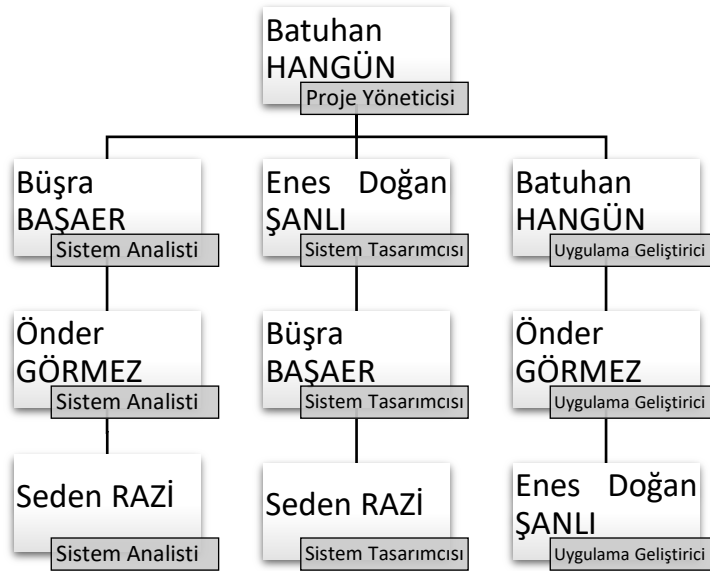
-Derslik adında yalnızca büyük harf girilebilmeli ve İngilizce karakter yer almalıdır. Kapasite bilgisi ise sadece tam sayı türünden olmalıdır.

1.3.Proje İş-Zaman Çizelgesi



Şekil 1:Dil Kursu Otomasyon Yazılım Projesi Gantt Diyagramı

1.4.Ekip Organizasyon Şeması



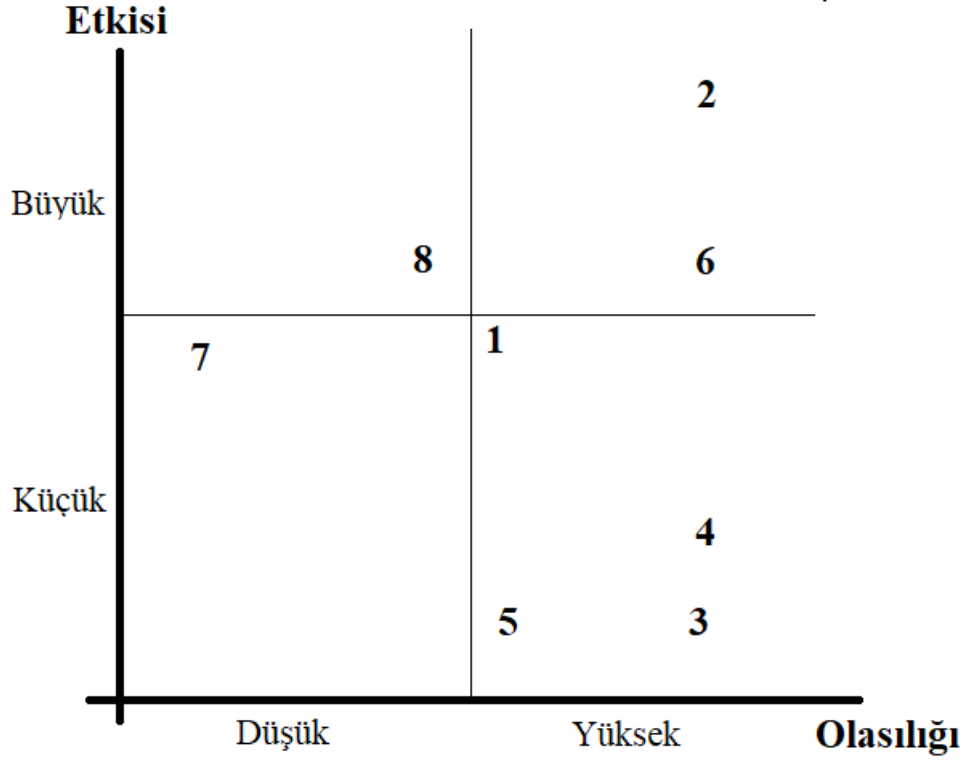
Şekil 2 :Dil Kursu Otomasyon Yazılım Projesi Organizasyon Şeması

1.5.Risk Analiz Tablosu

Proje kapsamında oluşabilecek risklere karşı proaktif yaklaşım, riskleri daha gerçekleşmeden önlemeye çalışmak, sergilenmiştir. Proje müşteri talebi ile gerçekleştirildiği için Pazar, satış gibi işletme riskleri bulunmamaktadır. “Bir Lisan Bir İnsan” dil kursu için üretilecek yazılım projesinde karşılaşılabilecek riskler Delphi Yöntemi ile belirlenmiştir. Ekip üyeleri ile belirlenen riskler Tablo1’de listelenmiş ve etki olasılık grafiği sonucuna göre bu risklerin bazıları için Risk Bilgi Sayfa’ları yazılmıştır.

Risk ID	Adı	Türü, Grubu	Etkisi	Olasılık
1	Teslim Tarihi	Proje	Orta	Orta
2	Proje Ekibi	Proje	Yüksek	Yüksek
3	Maliyet	Proje	Düşük	Yüksek
4	Uyarlanabilirlik	Teknik	Düşük	Yüksek
5	Kolay Kullanım	Teknik	Düşük	Orta
6	Ek İsterler	Proje	Orta	Yüksek
7	Metot,Araç Yetersizliği	Teknik	Orta	Düşük
8	Organizasyon	Proje	Orta	Orta

Tablo 1 :Dil Kursu Otomasyon Yazılım Projesi Risk Analizi Tablosu



Şekil 3:Dil Kursu Otomasyon Yazılım Projesi Risk Etki-Olasılık Grafiği

Risk 2#	Proje Ekibi		
Olasılık: Yüksek		Etki: Orta	Türü: Proje
Açıklama: Proje ekibinin iletişiminin zayıf olması ve iş planlamasında verilen görevin yerine getirilmemesi			
İşaretler:			
1. Ekip üyesinin iş yoğunluğu sebebiyle görevini aksatması 2. Toplantılara katılım gösterilmemesi. 3. Verilen işi yarım bırakma veya alanda yetkin olunmadığı için tamamlayamama 4. Birbirine bağımlı işlerin farklı yapılması. Use-Case Diyagramı dikkate alınmadan Kullanım Senaryosu yazılması.			
Önlemler			
1. Ekip içi iş dağılımı yapılırken ortak karar almak.			

- İş dağılımlarında ekip üyelerinin yokluğunun darboğaz oluşturmaması için iş parçasının minimum iki kişinin sorumluluğunda olması.
- Toplantı saatlerinin önceden planlanarak ekip üyelerinin aktif katılım ve uyumunun sağlanması

Tablo 2:Dil Kursu Otomasyon Yazılım Projesi Risk Bilgi Sayfası 1

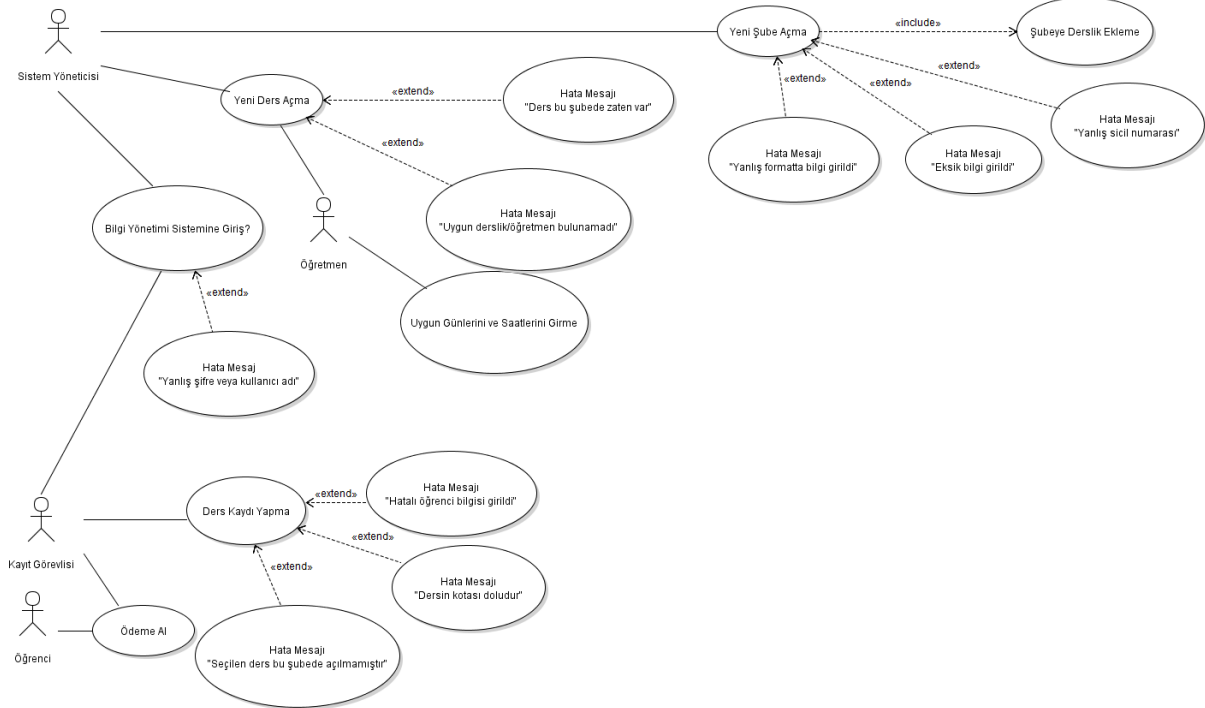
Risk 6#	Ek İsterler
Olasılık: Yüksek	Etki: Orta
Türü: Proje	
Açıklama: Müşterinin projede sık sık yeni istekte bulunması veya güncelleme istemesi	
İşaretler:	
<ol style="list-style-type: none"> Müşterinin ‘Buraya şu butonu da ekleyebilir miyiz?’ gibi proje başında kararlaştırılmamış isteklerin gelmesi. Anlaşmada gereği yapılacak raporlar için ‘Biz raporların tek Excel dosyasında farklı sayfalara basılmasını istiyoruz.’ gibi zaman ve maliyete sebep olacak isteklerde bulunulması. 	
Önlemler	
<ol style="list-style-type: none"> Proje başlamadan iş planı detaylandırılarak müşteri ile mutabakatname imzalanmalıdır. İş planında ucu açık cümleler netleştirilmelidir. Sözleşmede ek isterlerin ek ücrete tabi olacağı ve ekiple mutabık kalındığı takdirde yapılacağına dair açık beyan bulunmalıdır. 	

Tablo 3 :Dil Kursu Otomasyon Yazılım Projesi Risk Bilgi Sayfası 2

2. İSTEKLERİN MODELLENMESİ

Bu bölümde projemizin Kullanım Şeması ve Kullanım Senaryoları yer almaktadır.

2.1.Kullanım Şeması (Use-Case)



Şekil 4 :Dil Kursu Otomasyon Yazılımı Kullanım Şeması

2.2.Kullanım Senaryoları

Proje kapsamında ‘Bir Lisan Bir İnsan Dil Kursu’nun ihtiyaçları doğrultusunda Şube ve Derslik bilgisinin kontrolü için ‘Şube Derslik Bilgisinin Girilmesi’, gerekli kriterler sağlandığında yeni ders açılabilmesi için ‘Yeni Ders Açılması’ ve öğrencilerin ders kaydının kolaylıkla yapılabilmesi için ‘Ders Kaydı’ olarak 3 modül yapılmasına karar verilmiştir. Bahsedilen modüllerin kullanım senaryoları Tablo1, Tablo2 ve Tablo3’te gösterilmiştir.

2.2.1. Şube ve Derslik Bilgisi Girilmesi

Kullanım Senaryosu:	Şube ve derslik bilgisi girilmesi
Birincil Aktör:	Sistem yöneticisi
İlgililer ve Beklentileri:	İlgili 1: Sistem yöneticisi. İstedığı şube ve derslik bilgilerini hatasız bir şekilde girebilmeyi ve güncelleyebilmeyi bekler.
Ön Koşullar:	Yeni şube ve derslik açma talebi gelir.
Son Koşullar:	Şube ve derslik bilgisi girilir. Kayıt yapılır, işlem sonlandırılır.
Ana Senaryo:	1) Sistem yöneticisi sisteme giriş yapar. 2) Şubeye ait ad, sicil numarası, adres, ulaşım, iletişim, toplam derslik kapasitesi ve sosyal olanaklar bilgilerini girer. 3) Dersliğe ait ad ve kapasite bilgilerini girer. 4) Kaydet tuşuna basılır. 5) Kayıt işlemi sonlandırılır.

Alternatif Senaryo:	<p>1) Yanlış kullanıcı adı/şifre girilmiş ise</p> <ol style="list-style-type: none"> 1) "Yanlış kullanıcı adı/şifre girildi". 2) 1. adımın tekrar edilmesi beklenir. <p>2a) Geçersiz bir sicil numarası girilmiş ise</p> <ol style="list-style-type: none"> 1) "Geçersiz sicil numarası" mesajı gösterilir. 2) 2. adımın tekrar edilmesi beklenir. <p>2b) Herhangi bir bilgi boş bırakılmış ise</p> <ol style="list-style-type: none"> 1) İlgili bilgiye ait "Bilgi kısmı boş bırakılamaz" mesajı gösterilir. 2) 2. adımın tekrar edilmesi beklenir. <p>3a) Derslik bilgileri formata uygun şekilde girilmemiş ise.</p> <ol style="list-style-type: none"> 1) "Derslik bilgisi yanlış formatta girilmiştir" mesajı gösterilir. 2) 3. adımın tekrar edilmesi beklenir.
----------------------------	---

Şube Derslik Bilgisi Girilmesi Kullanım Senaryosu

2.2.2. Yeni Ders Açılması

Kullanım Senaryosu:	Yeni Ders Açılması
Birincil Aktör:	Sistem yöneticisi
İlgililer ve Beklentileri:	İlgili 1: Sistem yöneticisi. İsteddiği dersi uygun bir şubede açabilmeyi bekler.
Ön Koşullar:	Dersin açılması için yeterli sayıda talep gelmiş olmalıdır.
Son Koşullar:	Şube, derslik ve ders saati bilgisi girilir, uygun öğretmen seçilir. Ders açılır. işlem sonlandırılır.

Ana Senaryo:	1) Sistem yöneticisi sisteme giriş yapar. 2) Sistem yöneticisi dersin açılacağı şubeyi seçer. 3) Sistem yöneticisi açılacak dersin dilini seçer. 4) Sistem, sistem yöneticisine uygun derslikleri ve uygun öğretmenleri listeler. 5) Sistem yöneticisi dersin açılacağı dersliği seçer. 6) Sistem yöneticisi dersi verecek öğretmeni seçer. 7) "Dersi oluştur" tuşuna basılır. 8) Ders oluşturma işlemi sonlandırılır.
Alternatif Senaryo:	1) Yanlış kullanıcı adı/şifre girilmiş ise 1) "Yanlış kullanıcı adı/şifre girildi". 2) 1. adımın tekrar edilmesi beklenir. 2) Seçilen dile ait ders seçili şubede zaten açılmış ise 1) "Ders seçili şubede açılmış olarak görünmektedir" mesajı gösterilir. 2) İşlem sonlandırılır. 3) Uygun bir derslik ya da öğretmen yok ise 1) "Uygun derslik/öğretmen bulunmamaktadır" mesajı gösterilir. 2) İşlem sonlandırılır.

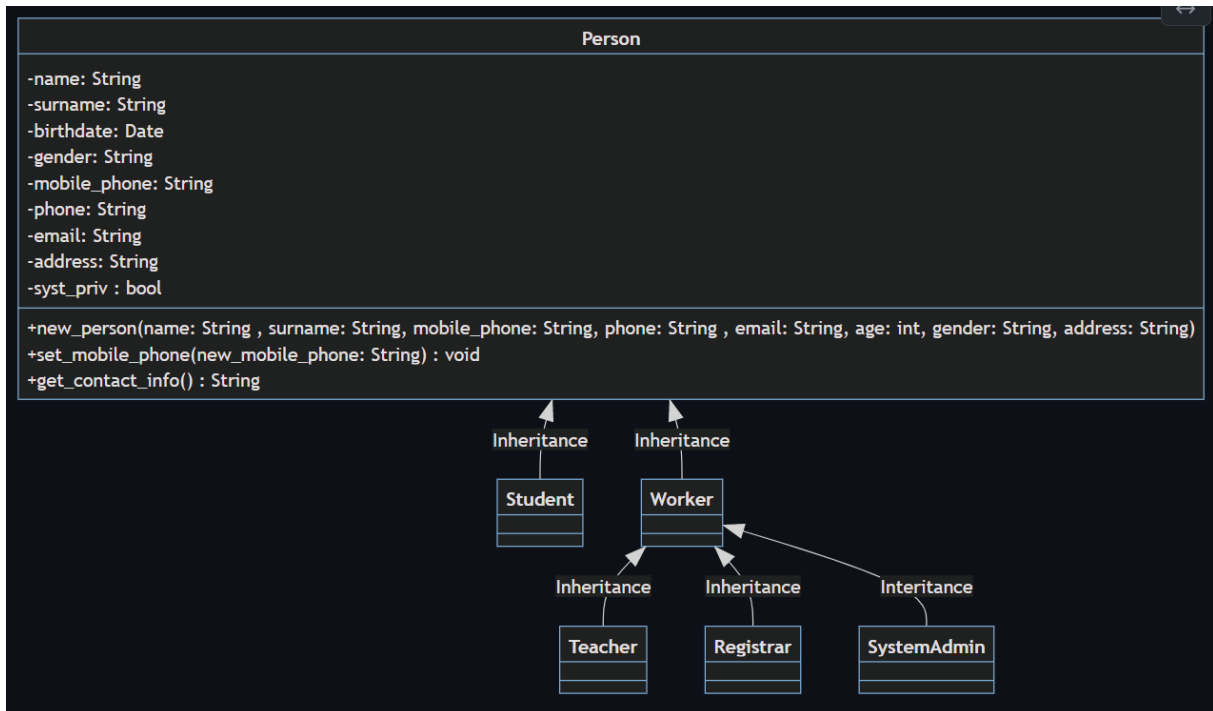
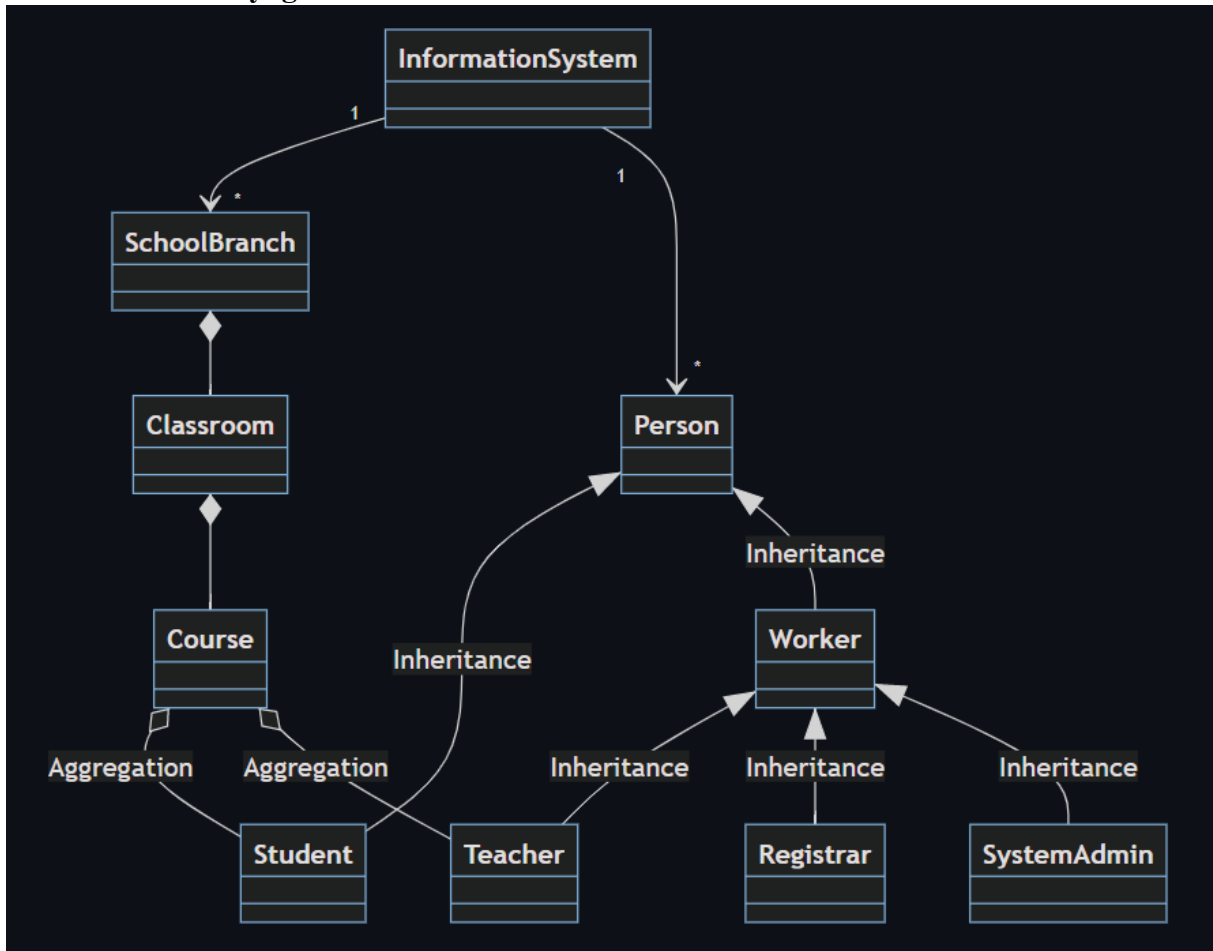
2.2.3. Öğrencinin Ders Kaydı

Kullanım Senaryosu:	Öğrencinin Ders Kaydı
Birincil Aktör:	Kayıt Görevlisi
İlgililer ve Beklentileri:	İlgili 1: Öğrenci. Ders kaydının başarılı bir şekilde tamamlanmış olmasını beklemektedir. İlgili 2: Kayıt Görevlisi: Sistemin kullanımının kolay olmasını beklemektedir.
Ön Koşullar:	Şube aktif olarak hizmet verebiliyor olmalıdır. Kapatılmış veya yeni açılacak bir şube olmamalıdır. Öğrencinin belgeleri tamamlanmış olmalı ve kayıt koşullarını sağlamalıdır. Şubede öğrencinin kayıt olmak istediği dersin açılmış olması gerekmektedir. Öğrencinin kayıt olmak istediği dersin kotası dolmamış olmalıdır.
Son Koşullar:	Öğrenci kaydının başarılı bir şekilde tamamlanması.
Ana Senaryo:	1) Kayıt görevlisi sisteme kullanıcı adı ve şifresi ile giriş yapar. 2) Kayıt görevlisi öğrenci modülüne girer. 3) Yeni öğrenci kaydı bölümüne girer.

	<p>4) Öğrencinin adı, soyadı, doğum tarihi, ev ve cep telefonları, e-posta adresi bilgileri sisteme girilir.</p> <p>5) Öğrencinin kayıt yaptırmak istediği kursa ait, kurs adı, kur ve kurs saati ile şube bilgilerini girilir.</p> <p>6) Ders kaydı tamamlanır, öğrenci bilgileri ekranına geri dönülür.</p>
Alternatif Senaryo:	<p>1a) Kayıt görevlisi yanlış kullanıcı adı veya şifre girmiş ise.</p> <p>1) Kayıt görevlisine "Hatalı şifre veya kullanıcı adı" mesajı gösterilir.</p> <p>2) 1. adımın tekrar etmesi beklenir.</p> <p>4a) Öğrencinin kişisel bilgilerinden biri hatalı girilmiş ise.</p> <p>1) "İlgili kullanıcı bilgisini yanlış ya da eksik girdiniz" mesajı gösterilir.</p> <p>2) 4. adımın tekrar edilmesi beklenir.</p> <p>5a) Kayıt yapılmak istenen kursun kotası dolmuş ise.</p> <p>1) "Kursa kaydolabilecek öğrenci sayısını aşmaktasınız" mesajı gösterilir.</p> <p>2) 5. adımın tekrar etmesi beklenir.</p> <p>5b) Kayıt yapılmak istenen kurs seçilen şubede açılmamış ise.</p> <p>1) "İlgili kurs seçilen şubede açılmamıştır" mesajı gösterilir.</p> <p>2) 5. adımın tekrar etmesi beklenir.</p>

3. NESNEYE DAYALI MODELLEME

3.1. UML Sınıf Diyagramı



Student
-courses :vector<String> -course_level :vector<String> -payment_infos :String
+new_student(person: Person, course_list: vector<String>, course_levels: vector<String>, payment_infos: String) : Student +get_course_list() +display_student() : void +delete_student() : bool

Worker
-start_date : String -end_date : String -active_worker : bool -salary : int -role : String
+new_worker(person: Person, start_date: String, end_date: String, active_worker : bool, salary: int, role: String) : Worker +delete_worker() : bool +update_salary(new_salary: double) : void +end_contract() : void

Teacher
-languages : vector<String> -available_branches : vector<SchoolBranch> -available_days : String -available_hours : String -available_times : Dictionary
+new_teacher(person: Person, languages : vector<String>, available_branches : vector<SchoolBranch>, available_days : String, available_hours : String) : Teacher +delete_teacher() : bool +set_available_times() +get_available_times() +find_teacher(teacher_list: vector<Teacher>, day: String)

Registrar
-user_name :String -password :String
+new_registrar() : Registrar +delete_registrar() : bool +log_in(syst_priv : bool, userName: String, password: String) : bool +log_out() : bool +add_to_course(course: Course) : bool +delete_from_course(course: Course) : bool +register_payment(student: Student, payment_info: student.payment_infos)

SystemAdmin

-user_name : String
-password : String

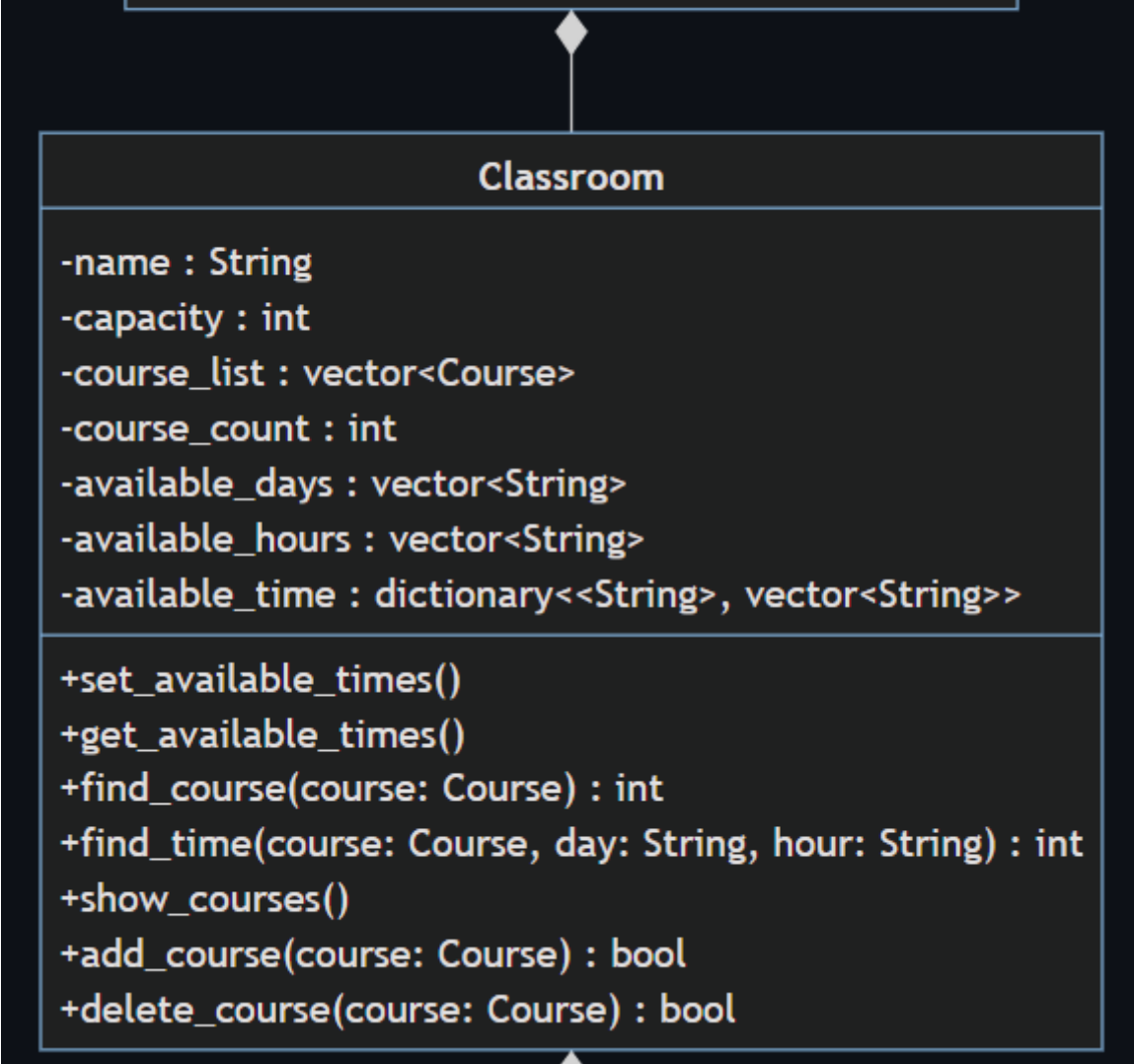
+new_system_admin() : SystemAdmin
+login_to_system()
+add_new_branch(new_branch: SchoolBranch)
+add_classroom(new_classroom: Classroom)
+add_new_course(new_course: Course)

SchoolBranch

-name : String
-id : String
-address : String
-public_transport : String
-private_transport : String
-social_benefits : vector<String>
-classroom_list : vector<ClassRoom>
-classroom_count : int

+show_classrooms()
+find_classroom(classroom: Classroom) : int
+add_classroom(classroom: Classroom) : bool
+delete_classroom(classroom: Classroom) : bool



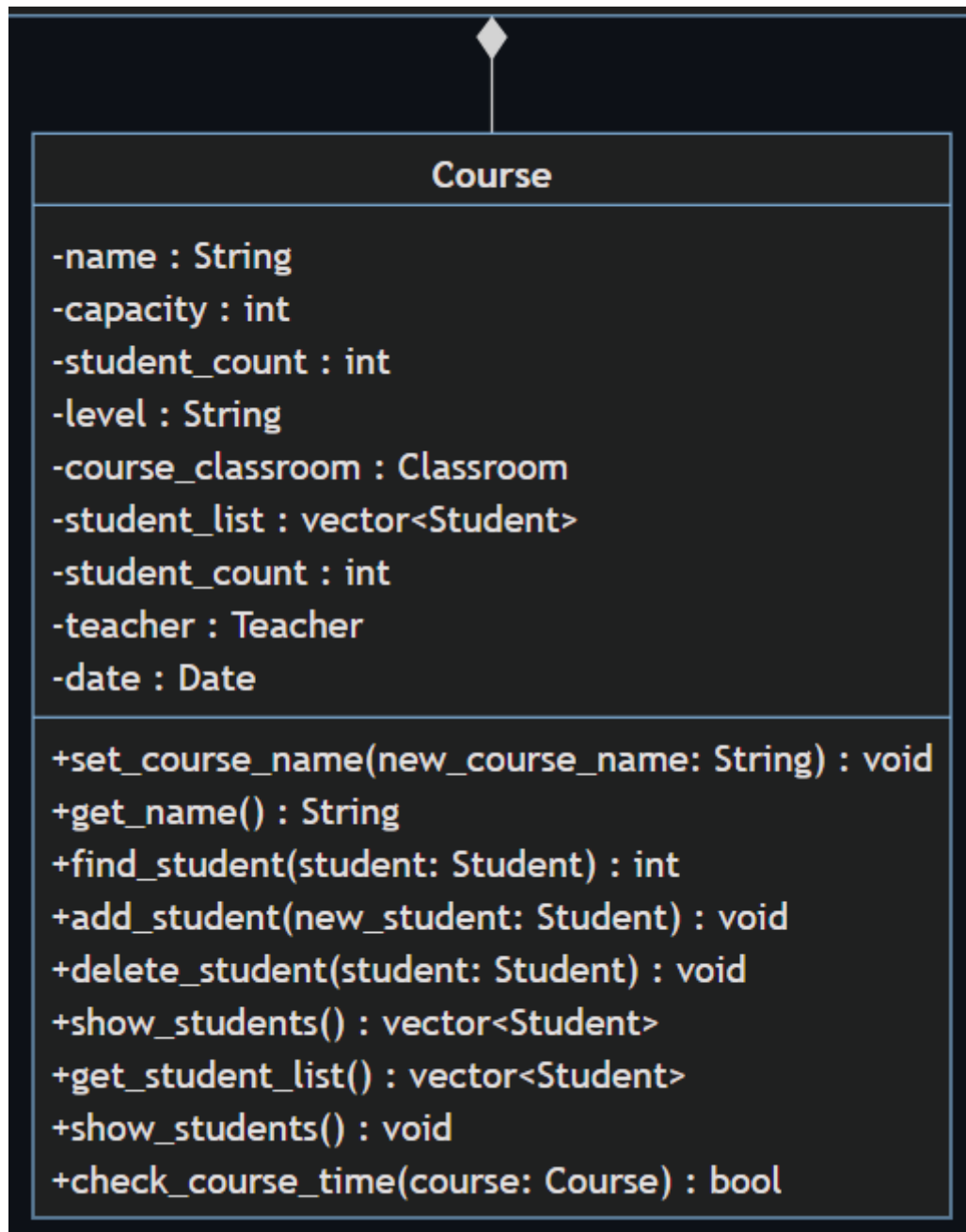


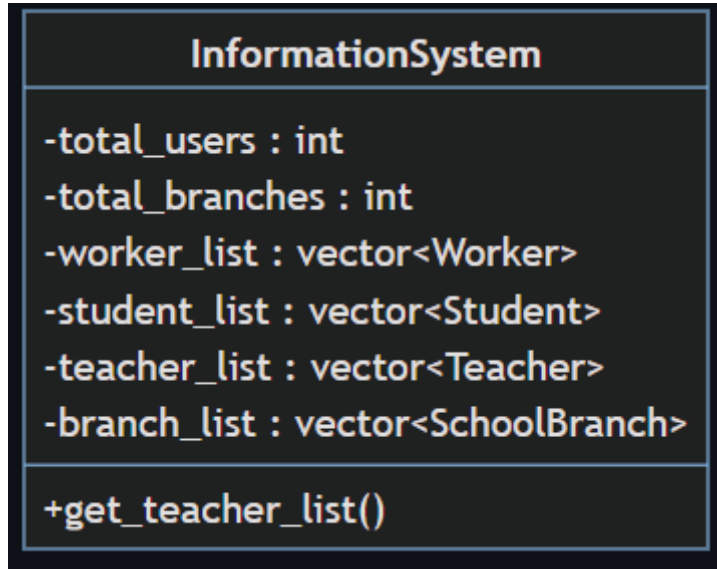
```
classDiagram
    class Classroom {
        -name : String
        -capacity : int
        -course_list : vector<Course>
        -course_count : int
        -available_days : vector<String>
        -available_hours : vector<String>
        -available_time : dictionary<<String>, vector<String>>
        +set_available_times()
        +get_available_times()
        +find_course(course: Course) : int
        +find_time(course: Course, day: String, hour: String) : int
        +show_courses()
        +add_course(course: Course) : bool
        +delete_course(course: Course) : bool
    }
    Classroom <|-- 
```

Classroom

-name : String
-capacity : int
-course_list : vector<Course>
-course_count : int
-available_days : vector<String>
-available_hours : vector<String>
-available_time : dictionary<<String>, vector<String>>

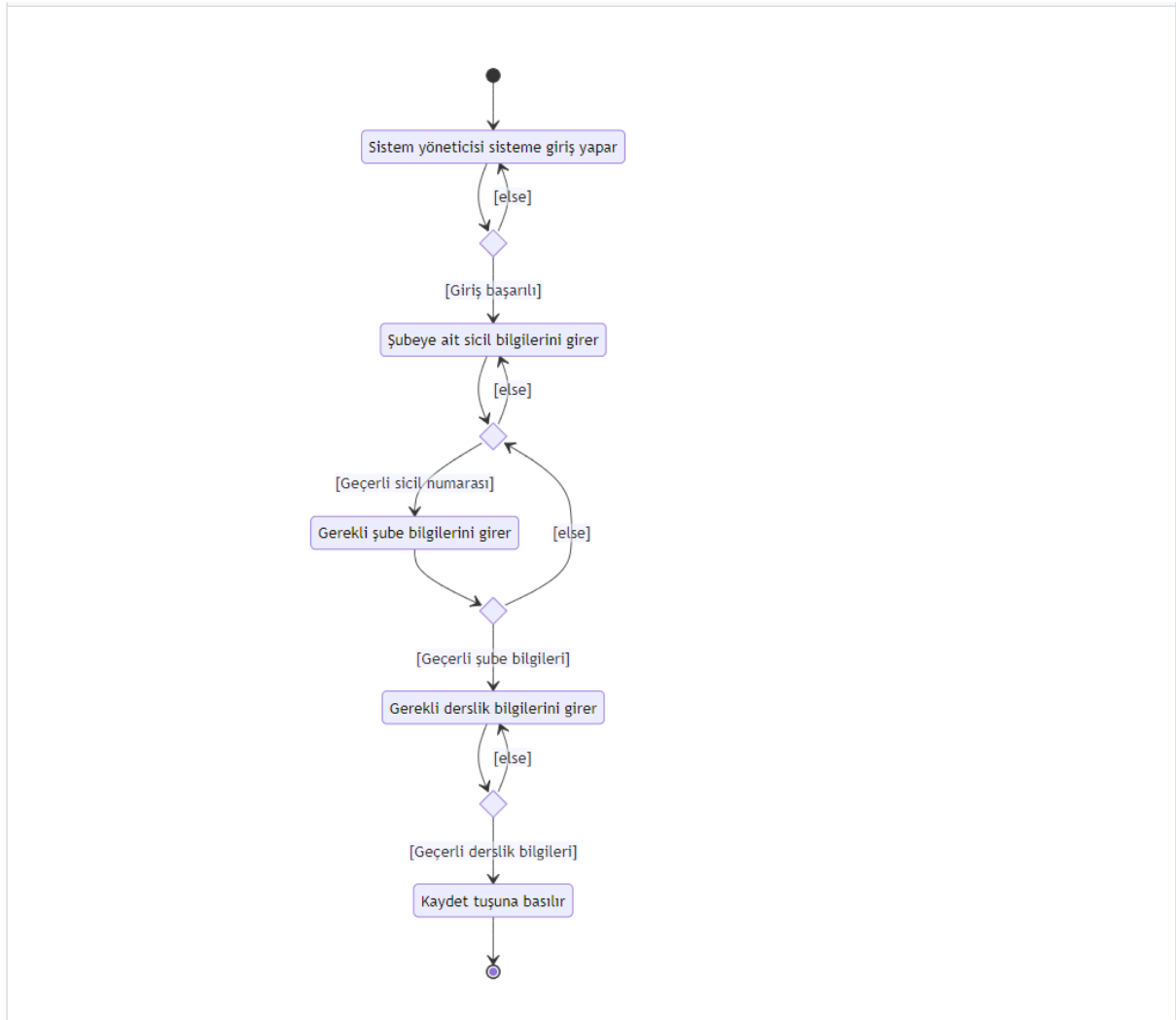
+set_available_times()
+get_available_times()
+find_course(course: Course) : int
+find_time(course: Course, day: String, hour: String) : int
+show_courses()
+add_course(course: Course) : bool
+delete_course(course: Course) : bool





4. NESNEYE YÖNELİK TASARIM DİYAGRAMLARI

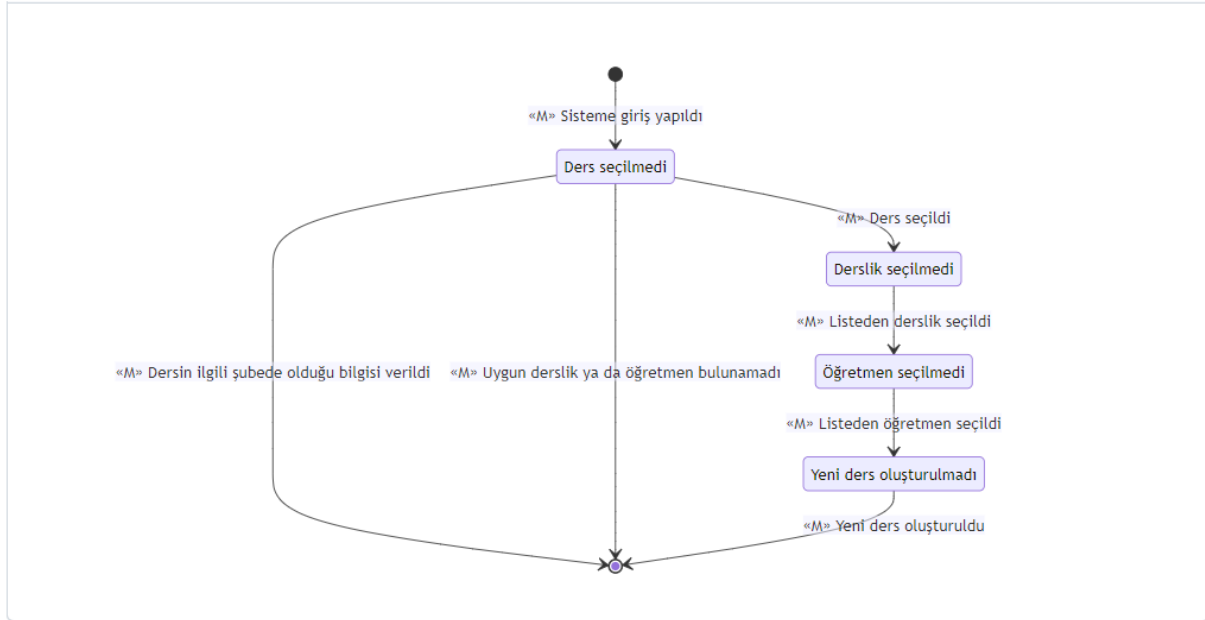
4.1.Etkinlik (Activity) Diyagramı



Şekil 5 :Şube ve derslik bilgisi girilmesi Etkinlik Diyagramı

Yeni şube/derslik açmak isteyen Sistem Yöneticisi kullanıcı adı ve şifresi ile sisteme giriş yapar. Girilen şifre veya kullanıcı adı hatalı ise giriş ekranında hata mesajı verilerek tekrar giriş yapması istenir. Başarılı bir şekilde giriş yapan kullanıcı varolan bir şubeye derslik için sicil numarasını girer. Girilen sicil numarası hatalı ise ekranda hata mesajı vererek tekrar girmesi istenir. Sicil numarası doğru girilen şubeye ait ad, adres, ulaşım, iletişim, toplam derslik kapasitesi ve sosyal olanaklar bilgilerini girerek kaydedilir. Kullanıcı bu şubeye derslik adında yalnızca büyük harf, İngilizce karakter yer alan ve kapasite bilgisi sadece tam sayı türünden olan dersliği ekleyebilir. Belirlenen şartlara uymayan kayıt için ekrana uyarı verilir ve tekrar girilmesi istenir. Şartlara uyan Derslik şubeye kaydedilir.

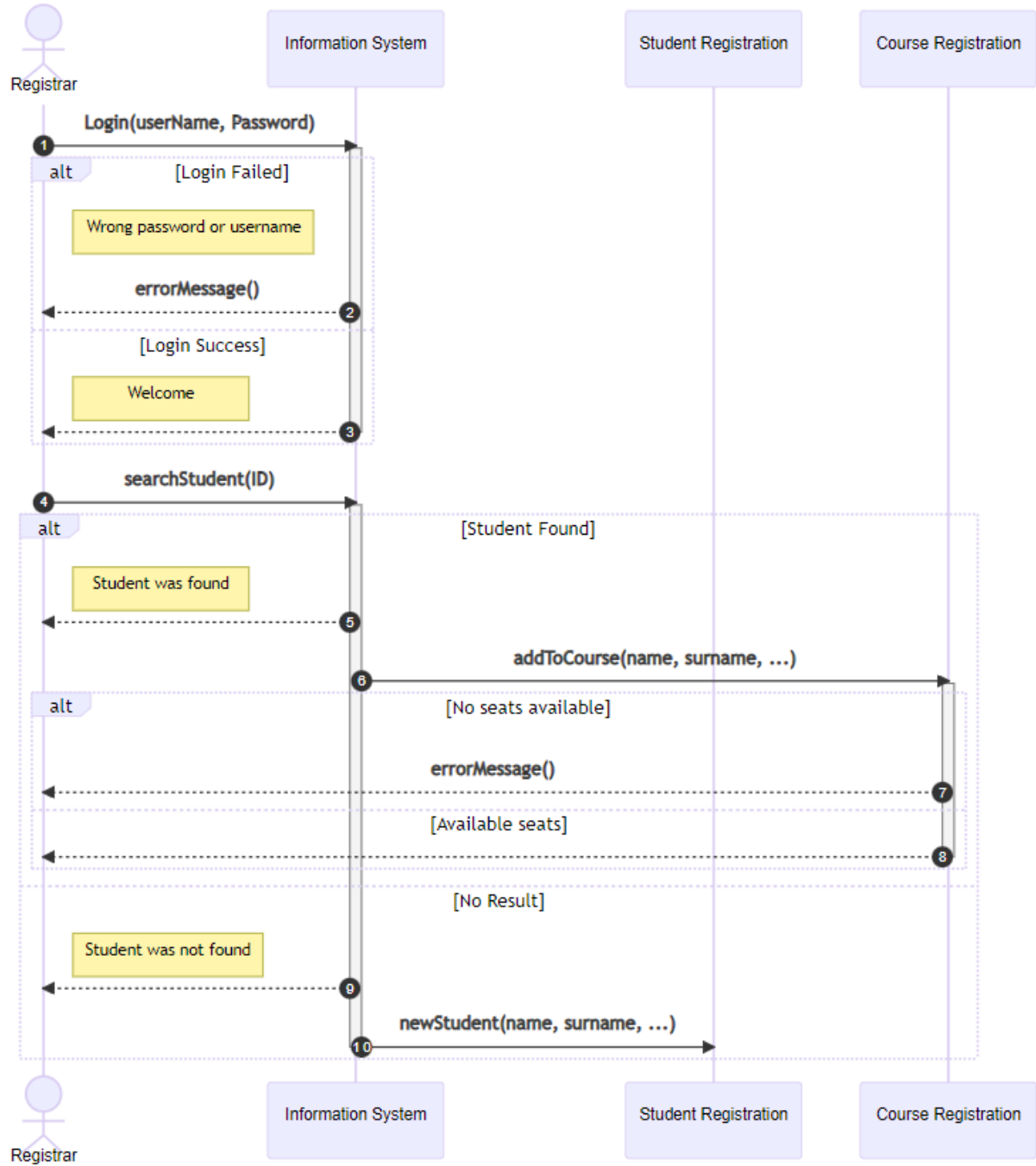
4.2.Durum (State) Diyagramı



Şekil 6 :Kaydı Yapılması Durum Diyagramı

Sisteme ders kaydetmek için öncelikle Sisteme giriş yapılır sonra Ders Seçimi için bekleme durumuna gelinir. Ders seçildiğinde seçilen ders ilgili şubede değilse veya ders için uygun derslik ya da öğretmen bulunamadıysa ekrana bilgi verilerek işlem sonlandırılır. Ders seçimi ile ilgili sorun yoksa Derslik Seçimi Durumuna gelinir. Listedeki derslik seçildiğinde öğretmen seçimi durumuna gelinir. Listedeki öğretmen de seçildiğinde yeni ders oluşması için sistem beklenir ve ders oluşturulur.

4.3.Ardışıl (Sequence) Diyagram



Şekil 7 :Öğrencinin Ders Kaydı Ardışıl Diyagramı

Öğrenci için ders kaydı yapmak isteyen Kayıt Görevlisi sisteme girmek için kullanıcı adı ve şifresini girer. Şifre hatalı ise giriş başarısız olur ve ekrana ‘Wrong password or username’ mesajı gelir. Sisteme girişi başarılı olduğunda ekrana ‘Welcome’ mesajı gelir. Ders kaydolmak istenen öğrenci sistemde aratılır Öğrenci kaydı bulunamadıysa ekrana ‘Student was not found’ uyarısı gelir. Aranılan öğrenci kaydı bulunduysa öğrenci için ders seçilir. Seçilen kursta

kontenjan kalmamış olması durumunda ekrana ‘No seats available’ uyarısı gelir.Kursta kontenjan olması durumunda öğrencinin ders kaydı tamamlanır .

5. İZLENEBİLİRLİK TABLOSU

Gereksinimler/Bakış Açıları	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11
G1	X										
G2	X	X									
G3	X	X									
G4	X	X				X				X	X
G5	X		X							X	
G6	X					X					
G7	X						X				
G8	X							X			
G9									X		

G1	Kayıtların saklanması
G2	Şubeler ile ilgili işlem
G3	Yeni şube ve derslik girdisi işlemi
G4	Yeni ders açılması işlemi
G5	Öğrencinin ders kaydının yapılması
G6	Şube bilgilerinin saklanması
G7	Öğretmenler hakkında bilgiler
G8	Öğrenciler hakkında bilgiler
G9	Ödeme tahsilatı

B1	Information System
B2	SystemAdmin Class
B3	SystemAdmin Class
B4	SystemAdmin Class
B5	Registrar Class
B6	SchoolBranch Class
B7	Teacher Class
B8	Student Class
B9	Registrar Class
B10	Course Class
B11	Classroom Class

6. BİRİM TESTLER

```
from person import Person
from student import Student
from worker import Worker
from teacher import Teacher
from course import Course
from classroom import Classroom
from registrar import Registrar
from systemAdmin import SystemAdmin
from school_branch import SchoolBranch
from information_system import InformationSystem
import datetime
import unittest

class TestId(unittest.TestCase):

    @classmethod
    def setUpClass(cls):
        print("setUpClass ")

    @classmethod
    def tearDownClass(cls):
        print("tearDownClass")

    def setUp(self):
        print("setUp")

    # Öğrenci nesnesi oluşturma
    self.person1 = Person('Batuhan', 'Hangün', '1111111',
datetime.date(
        1991, 8, 19), 'Male', '05336833244', '02129854478',
'batuhan_hangun@mymail.com', '59/5, Maslak, İstanbul')
    self.student1 = Student(
        self.person1, ('English', 'German'), ('A1', 'B1'), (4, 5))

    # Öğretmen nesnesi oluşturma
    self.person2 = Person('Gregory', 'Ruslanov', '12345546554',
datetime.date(1991, 3, 11), 'Male',
        '05335433244', '02129865478',
'ruslanov_russian@mymail.com', '21/3, Bebek, İstanbul')
```

```

        self.worker1 = Worker(self.person2, datetime.date(2010, 3,
12), True, 30000, 'Teacher')
        self.teacher1 = Teacher(self.person2, self.worker1,
["Russian"],
                                ["Şişli", "Levent"], ["Monday",
"Thursday", "Friday"], [{"15:00", "16:00"}, {"12:00", "14:00"},
{"09:00"}])
        self.teacher1.set_available_times()

# Kayıt Görevlisi nesnesi oluşturma
        self.person3 = Person('Önder', 'Görmez', '3333333',
datetime.date(1993, 2, 21), 'Male',
                                '051236653244', '02129544878',
'onder_gormez@mymail.com', '66/2, Şişli, İstanbul')
        self.worker2 = Worker(self.person3, datetime.date(2010, 3,
12), True, 40000, 'Registrar')
        self.registrar1 = Registrar(self.person3, self.worker2,
"ogormez", "gormez*123-*")

# Admin nesnesi oluşturma
        self.person4 = Person('Samuel', 'Jackson', '3333333',
datetime.date(1960, 12, 21), 'Male',
                                '051236653244', '02129544878',
'SamuelJackson@mymail.com', '66/2, Şişli, İstanbul')
        self.worker3 = Worker(self.person4, datetime.date(2010, 3,
12), True, 40000, 'Admin')
        self.sysadmin1 = SystemAdmin(
            self.person4, self.worker3, "jack_s", "bad*546-mf")

# Classroom nesnesi oluşturma
        self.classroom1 = Classroom(
            "A134", 32, ["Monday", "Friday"], [{"15:00", "17:00"},
{"09:00"}])
        self.classroom1.set_available_times()

# Şube nesnesi oluşturma ve class ekleme
        self.branch1 = SchoolBranch(
            "Maslak", "543463", "Ayazağa, 34/2", "41AT, 500T, 41ST",
["E6 Otoyolu", "E5 Otoyolu"], ["Langırt", "PS5"])
        self.branch1.add_classroom(self.classroom1)

```

```

# Course nesnesi oluşturma
self.course1 = Course("Russian", 32, level = "A1")

# Information System nesnesi oluşturma
self.information_system1 = InformationSystem(
    10, 4)

# Oluşturulan Information System nesnese öğretmen ekleme
self.information_system1.teacher_list.append(self.teacher1)

def tearDown(self):
    Enes Doğan Şanlı
    print("tearDown\n")

def test_update_salary(self):
    Enes Doğan Şanlı
    print("test_update_salary")
    self.assertEqual(self.worker1.update_salary(40000), 40000)

def test_update_id(self):
    Batuhan Hangün
    print("test_update_id")
    self.assertEqual(self.student1.update_id("1111112"), "1111112")
")

def test_coursename_update(self):
    Batuhan Hangün
    print("test_coursename_update")
    self.course1.set_course_name("English")
    self.assertEqual(self.course1.name, "English")

def test_set_mobile_phone(self):
    Önder Görmez
    print("test_set_mobile_phone")
    self.person1.set_mobile_phone("05336833243")
    self.assertEqual(self.person1.mobile_phone, "05336833243")

def test_get_contact_info(self):
    Önder Görmez
    print("test_get_contact_info")
    self.assertEqual(self.person1.get_contact_info(), ("053368332
44", "02129854478", "batuhan_hangun@mymail.com", '59/5, Maslak,
İstanbul'))

def test_update_email(self):
    Büşra Başaer
    print("test_update_email")
    self.assertEqual(self.student1.update_email("batuhanhangun@m
ymail.com"), "batuhanhangun@mymail.com")

```

```

def test_get_course_list(self):
    print("test_get_course_list")
    tuple_len = len(self.student1.courses)
    self.assertEqual(self.student1.get_course_list(), (self.student1.courses, tuple_len))

def test_find_teacher(self):
    print("test_find_teacher")
    self.assertEqual(self.teacher1.find_teacher(self.information_system1.get_teacher_list(), "Monday"), self.information_system1.get_teacher_list())

def test_update_name(self):
    print("test_update_name")
    self.assertEqual(self.student1.update_name("Dogan"), "Dogan")

def test_get_teacher_list(self):
    print("test_get_teacher_list")
    self.assertEqual(self.information_system1.get_teacher_list(), self.information_system1.teacher_list)

if __name__ == '__main__':
    unittest.main()

```

```
.tearDownClass
```

```
-----
Ran 10 tests in 0.025s
```

```
OK
```

KAYNAKÇA

1. [Mermaid Documentation](#)
2. [Python Documentation](#)