

STAJ DEFTERİ

ÖĞRENCİ

ADI SOYADI : Maya Karah Bala

NUMARASI : 1521221114

TARİH :

ÖĞRENCİ BİLGİLERİ

Öğrenci No	1521221114
Öğrenci Adı ve Soyadı	Maya Karah Bala
Doğum Yeri ve Tarihi	Suriye 14/05/1996



STAJ Y APILAN KURULUŞ/FİRMA

Adı	isbak
Adresi	Seyrantepe Mah. Cendere Caddesi No:56 Kağıthane / İSTANBUL
Başlangıç Tarihi	18/06/2018
Bitiş Tarihi	30/07/2018

SORUMLU KURUM YETKİLİSİ

Adı ve Soyadı	
Ünvanı	
Onayı	

BİTİŞ ONAYI





Staja Sayılan Gün	
Staj Komisyonu Onayı	
Bölüm Başkanlığı Onayı	

Yaprak No: 1	Kısım:
Tarih: 18/06/2018	Yapılan İş:
<p>Bugün stajım ilk günü, şirkete geldiğimde evraklarımı teslim etmek için insan kaynaklarına gittim. Evraklarda hastane raporları eksik olduğu için buradan çıkıp hastaneye gittim ve gereken testleri yaptırdım. Sonra İsbak'taki hekim raporları kontrol etti ve onayladı. Hekimin onayladığı evrağı daha sonra İş Sağlığı ve Güvenliği Uzmanı da onayladı ve son olarak Uygulama Mühendisliği Birim Şefi Furkan ÇELEBİ onayladı ve evrağı İnsan Kaynaklarına teslim ettim. Sonrasında staj yapacağım birime geçip oradaki çalışanlarla tanıştım ve stajın nasıl geçeceği, isbak genel olarak nelerle uğraştığı ile ilgili konuştuk.</p> <p>İsbak Hakkında Genel Bilgiler:</p> <p>İstanbul Bilişim ve Akıllı Kent Teknolojileri A.Ş. (İSBK), İstanbul Büyükşehir Belediyesi (İBB) tarafından trafik ve sistem mühendisliği ile projelendirme ve uygulama hizmetlerini gerçekleştirmek amacıyla 1986 yılında kurulmuştur.</p> <p>Faaliyet Alanları</p> <ul style="list-style-type: none">• Akıllı Şehir Teknoloji ve Uygulamaları• Ulusal Ve Uluslararası Danışmanlık Ve Planlama Hizmetleri• Akıllı Ulaşım Sistemleri• Ulaşım Planlama ve Coğrafi Bilgi Sistemleri• Haberleşme-Görüntü ve Kent Güvenlik Yönetim Sistemi• Akıllı Aydınlatma Sistemi• Tünel Yönetim Sistemi• Filo Yönetim Sistemi	
Kontrol Eden:	Onay

Yayın Tarihi: 08/03/2017

Yaprak No: 3	Kısım:
Tarih: 20/06/2018	Yapılan İş: CoDeSys KURULUMU VE ST KULLANIMI
<p>CoDeSys IDE'yi indirdim ve IDE hakkında genel bir bilgi elde ettikten sonra Structured Text (ST) hakkında ayrıntılı bir araştırma yaptım. Dilin genel yapısı ve syntaxı hakkında bilgi topladım ve bildiğim programlama dilleriyle karşılaştırdım</p> <p>Dilin bazı özellikleri:</p> <ul style="list-style-type: none">• Atama bu şekilde yapılır A:=B;• Eşit değilse <> bu sembollerle gösterilir <p>Yaptığım örnekler:</p> <p>1- (şart ifadesi)</p> <pre>IF temp<17 THEN heating_on := TRUE; ELSE heating_on := FALSE; END_IF;</pre> <p>2- (for ifadesi)</p> <pre>FOR Counter:=1 TO 5 BY 1 DO Var1:=Var1*2; END_FOR;</pre> <p>3- (while ifadesi)</p> <pre>WHILE counter<>0 DO Var1 := Var1*2; Counter := Counter-1; END_WHILE</pre> <p>4-(REPEAT loop)</p> <pre>Do while gibi çalışır REPEAT Var1 := Var1*2; Counter := Counter-1; UNTIL Counter=0 END_REPEAT;</pre>	
Kontrol Eden:	Onay

Yaprak No: 4	Kısım:
Tarih: 21/06/2018	Yapılan İş: TÜNEL YÖNETİM YAZILIMI
<p>Bugün Necati Bey üzerinde çalıştığı projeyi bize anlattı. Proje İstanbul'daki tüneller içinde yapılan işlemlerden oluşuyor.</p> <p>Projede kullanılan yazılım özel bir yazılım. Scriptler Visual Basic veya C# ile yazılabiliyor. Programın desteklediği bazı özelliklerden biri, bazı kısımlarda görsel arayüz sunuyor bu arayüzden seçtiği ayarlara göre arka planda kod üretiyor.</p> <p>Kısaca sistem, tünel içerisindeki tüm sistemlerin verilerinin birbirileri arasındaki değiş tokuşu ve gerekli işlemlerin otomatik ya da kullanıcı komutları ile yerine getirilmesini sağlar</p> <p>Tünel Yönetim Sistemleri</p> <p>Aydınlatma</p> <ul style="list-style-type: none">Tünel içi aydınlatmasının dış ortamın ışık şiddetine bağlı olarak otomatik ayarlanmasıOtomatik ve manuel modda tünel içi aydınlatma seçenekleri Otomatik: örnek olarak sabah saatlerinde otomatik olarak ışıklar azaltılır, Manuel: sisteme giriş yaparak istediğin zamanda ışıkları söndürüp yakabilirsin.Gün ışığına bağlı olarak operatörün farklı kademelerde tünel içi aydınlatma seçme imkânı <p>Yangın sistemi</p> <ul style="list-style-type: none">Yangın butonları ve sensörleriyle noktasal yangın algılamaDoğrusal yangın algılama sistemiyle tünel içi sıcaklık ölçümü ve yangın algılamaYangın olduğu zamanlarda sistem otomatik çalışır. <p>Havalandırma</p> <ul style="list-style-type: none">Otomatik ve manuel modda çalışabilen tünel içi havalandırma otomasyonu.Jetfanların eş yaşlandırma prensibine göre çalıştırılabilmesi. (Yeni olanlardan başlayarak eski jetfanlara doğru çalıştırılır) <p>Alarm</p> <ul style="list-style-type: none">Herhangi bir sistemden gelen alarmların anlık izlenebilmesi ve veri tabanına kaydedilmesiMevcut alarmların aktif kullanıcı tarafından onaylanmasıGeçmişe dönük her türlü alarmların kolayca sorgulanabilmesiAlarmların istenilen kriterlere göre filtrelenebilmesi <p>Kameralar</p> <p>Tünel içinde tüneli kapsayacak şekilde birden fazla kamera var, kameralar hareketli istediğin yönde hareket ettirebiliyorsun ve bütün video ve görüntüler veri tabanda kaydediliyor.</p>	
Kontrol Eden:	Onay

Yaprak No: 5	Kısım:
Tarih: 22/06/2018	Yapılan İş: GÜVENLİK KAMERA SİSTEMLER
<p>Belediyenin güvenlik sistemlerin çoğu kameralara dayanıyor. Sistemin ihtiyaçları tamamen karşılaması ve en uygun maliyetle kurulabilmesi için sistemde kullanılacak bileşenlerin ve teknolojinin çok iyi seçilmesi gerekir.</p> <p>Kullanılan Teknolojiye Göre Kameralar</p> <p>1- Analog Kameralar</p> <p>Pazarda en fazla karşılaşılan ve daha eski teknolojiye sahip ürünlerdir. Montajının kolay ve maliyetlerinin düşük, çekilen görüntü UTP kablolar üzerinden kayıt cihazına (DVR) iletilir. Kayıt cihazı bu görüntüleri dijitale çevirip yüksek kalitede saklar.</p> <p>2- IP Kameralar</p> <p>IP kamera tıpkı bilgisayar gibi doğrudan bir ağa bağlıdır. Video direkt olarak dijital ortama kaydedildiği için onu dijitalleştirmek için bir DVR'a ihtiyacınız olmayacaktır. Kamerayı bir kayıt cihazına bağlamanıza gerek kalmaz, çünkü her cihaz networke bağlıdır</p> <div style="display: flex; justify-content: space-around; align-items: center;"><div style="text-align: center;"> IP Kamera</div><div style="text-align: center;"> Analog Kamera</div></div> <p>Kayıt Cihazları</p> <p>Analog olarak kullanılan kayıt cihazı DVR olarak adlandırılır. Dijital olarak kullanılan sistemlerde network tabanlı IP kameralardan gelen görüntüler NVR ve yazılım tabanlı server üzerinde çalışan yazılımlar vasıtasıyla kayıt yapılır</p> <div style="display: flex; justify-content: space-around; align-items: center;"><div style="text-align: center;"> NVR</div><div style="text-align: center;"> DVR</div></div>	
Kontrol Eden:	Onay

Yaprak No: 6	Kısım:
Tarih: 25/06/2018	Yapılan İş: YÜZ TANIMA VE ALGILAMA SİSTEMİ
<p>Uygun kamera ve kayıt cihazı seçtikten sonra topladığımız verileri analiz ederek çok amaçlı sistemler gerçekleştirilebiliriz.</p> <p>Yüz tanınması gerçekleştirilecek personelin yüzü cihaz üzerinde bulunan yüksek çözünürlüğe sahip kameralar ile sisteme 3 boyutlu olarak tanımlanır. 3 boyutlu resim üzerinde referans noktalar belirlenir ve yüz tanıma algoritması ile sayısal verilere dönüştürülür.</p> <p>Biometrik yüz haritası olarak adlandırabileceğimiz bu algoritma; burunun gözlere oranı, gözlerin birbirine oranı, gözlerin çeneye oranı gibi basit şekilde anlatabileceğimiz şekilde referans noktaların milimetrik olarak birbirine uzaklığını ölçer. Bu referans noktalar yani oranlar bir insanın parmak izi gibidir sadece o insana özeldir ve aynısı yoktur. İnsan gözüyle fark edilemeyecek bu özellikler yüz tanıma cihazı sayesinde benzersiz bir biometrik tanımlama sistemine dönüşmektedir.</p> <div data-bbox="531 911 1056 1207" data-label="Image"></div> <p>Özetle yüz tanıma sistemi bir dijital video kamera ile bir kişinin yüz görüntülerini analiz eder. Gözler, burun, ağız ve çene kenarlarındaki mesafeler de dahil olmak üzere bütün yüz yapısını ölçer. Bu ölçümler bir veritabanında saklanır ve bir kullanıcı kamera önüne geldiği zaman yapılacak karşılaştırmalar için kullanılır.</p> <p>Kullanıldığı alanlar</p> <p>Güvenlik sistemleri :</p> <p>Suçluları takip etmek</p> <p>Personel takip sistemi :</p> <p>Personelinin giriş çıkış saatlerini otomatik olarak kayıt altına alınabilir</p> <p>Öğrenci takip sistemi :</p>	
Kontrol Eden:	Onay

Yaprak No: 7	Kısım:
Tarih: 26/06/2018	Yapılan İş: YOLCU SAYMA SİSTEMİ
<p>Bugün Şadi Bey bize yolcu sayma sistemi hakkında bilgiler verdi. Toplu taşıma araçlarında İsbak tarafından gerçekleştirilen bir projedir, genel olarak projenin in ve out counterleri var kişinin hareket yönü tespit edildikten sonra hareket yönü aracın dışından içerisine doğru ise kişi bindi demek in sayacı artıracak ters yönde ise kişi indi out sayacı artıracak sonra bu sayaçları birbirinden çıkarttırarak her durakta araçta kaç kişi kaldığını hesaplanır.</p> 	
<p>Yolcu Sayma Sistemi ile elde edilen veriler, Merkezi Yönetim ve Raporlama yazılımında işlenerek, çok amaçlı raporlar elde edilebilir. Bu raporlar ve göstergeler Toplu Taşıma Hizmet Birimleri tarafından anlık olarak web üzerinden de takip edilebilmektedir.</p> <ul style="list-style-type: none">-Hat Bazında Yolcu Yoğunluk Raporu-Sefer Bazında Yolcu Yoğunluk Raporu-Durak/İstasyon Bazında Yolcu Yoğunluk Raporu-Araç (Otobüs/Tramvay/Metro Vagonu) Bazında Yolcu Yoğunluk Raporu; <p>Alışveriş merkezlerinde de kişi sayma sistemi kullanılıyor. Sistem genel olarak kamera kayıtlarından alınan video kayıtlarına dayanıyor, amaç mağazaların performansını artırmak ve müşteri kazanmak. Bunun için mağazaya kaç kişi girdi, giren kişi kayıtlı bir müşteri mi, mağazadan bir şey satın almadan kaç kişi çıktı, ona göre müşterilerin davranışlarını analiz ederek en iyi şekilde hizmet vermeye çalışıyorlar.</p>	
Kontrol Eden:	Onay

Yaprak No: 8	Kısım:
Tarih: 27/06/2018	Yapılan İş: SOSYAL PROJELER
<p>Akıllı Konteyner (Akıllı Geri Dönüşüm) Geri dönüşebilir katı atık toplama otomatı (akıllı konteyner), ödüllendirme yöntemiyle geri dönüşüm bilincini aşılama ve bu bağlamda elde edilecek geri kazanımla ülke ekonomisine ve doğal kaynaklara katkı sağlamak amacıyla geliştirilmiştir.</p> <p>Hikayematik Otobüs, tramvay, metrobüs, metro gibi toplu ulaşım duraklarına kurulan sistem ile bekleyen yolcular için bekleme süresince okuyabilecekleri hikaye çıktısı vermek ve toplumda okuma alışkanlığını yaygınlaştırmak amacıyla geliştirilmiştir.</p> <p>Sistemin Sahip Olacağı Özellikler: Dokunmatik renkli ekran İstanbul Kart okuyucu, kartın okutulması ile hikaye alabilme Termal yazıcı ile çıktının hızlı şekilde verilmesi 1 dk, 3dk, 5 dk uzunluklarında hikaye yazma seçeneği Farklı dillerde hikaye yazma seçeneği Farklı yaş gurupları için hikaye yazma seçeneği Merkez bağlantısı ile gün içerisinde aynı kişiye farklı hikayeler verme İstanbul kart ile aynı kişiye belli sayıda hikaye verme.</p> <div></div> <div></div> <p>Hikayematik Akıllı Konteyner</p> <p>Uygulama Mühendisliği Şefliği Showroom</p>	
Kontrol Eden:	Onay

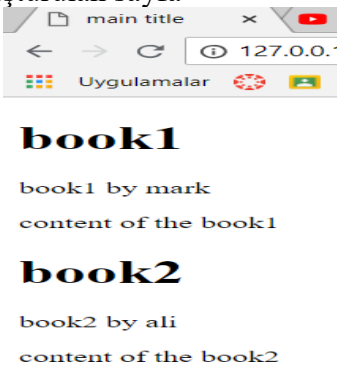
Yaprak No: 9	Kısım:
Tarih:28/06/2018	Yapılan İş: Python'a GİRİŞ
<p>String Kullanma Yöntemleri: Python'da formatlı string birkaç yolla nasıl yapıldığını öğrendim resimdeki tüm örnekler aynı sonuç verir</p> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <pre>s="maya" print(f"hello ,{s}")</pre> <p>hello ,maya</p> <hr/> <pre>print("hello ,{s}")</pre> <p>hello ,{s}</p> </div> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <pre>print("hello ,{}".format(s))</pre> <p>hello ,maya</p> <hr/> <pre>print("hello ",s)</pre> <p>hello maya</p> </div> </div> <p>Time Modülü: Time modülü kullanarak bekletme nasıl yapıldığını öğrendim. Bu döngüde saniyede bir sayı yazdırılıyor.</p> <pre>from time import sleep for i in range(100): print(i,end=" ") sleep(1)</pre> <p style="text-align: center;">0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</p> <p>Python'da == Sembolu Python'da == sembolü stringin içeriği kontrol etmek için kullanıyor java ise adres karşılaştırması için kullanılır</p> <div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <pre>s1="string" s2="string" if s1==s2: print("same")</pre> <p style="text-align: center;">same</p> <p style="text-align: center;">Python</p> </div> <div style="border: 1px solid #ccc; padding: 5px; width: 45%;"> <pre>String s1 =new String("string"); String s2 =new String("string"); if (s1 == s2) { System.out.println("same"); } else System.out.println("not same");</pre> <p style="text-align: center;">not same</p> <p style="text-align: center;">java</p> </div> </div> <p>Multiple Atama Python çoklu (multiple) atamayı destekliyor. Örnek takas (swap) işlemi yapmak için ayrı bir değişkene ihtiyacımız yok.</p> <pre>x=8 y=9 print(f" x is {x} and y is {y}") x,y=y,x print(f" x is {x} and y is {y}")</pre> <p style="text-align: center;">x is 8 and y is 9 x is 9 and y is 8</p>	
Kontrol Eden:	Onay

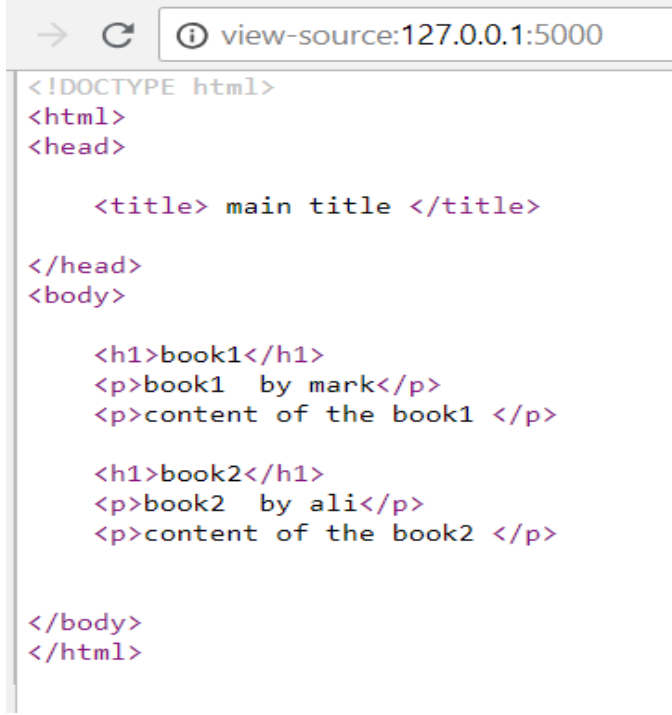
Yaprak No: 10	Kısım:
Tarih: 29/06/2018	Yapılan İş: PROBLEM ÇÖZME
<p>Verilen problem:</p> <p>Given an array a that contains only numbers in the range from 1 to a.length, find the first duplicate number for which the second occurrence has the minimal index. In other words, if there are more than 1 duplicated numbers, return the number for which the second occurrence has a smaller index than the second occurrence of the other number does. If there are no such elements, return -1.</p> <ul style="list-style-type: none">For a = [2, 1, 3, 5, 3, 2], the output should be firstDuplicate(a) = 3. There are 2 duplicates: numbers 2 and 3. The second occurrence of 3 has a smaller index than the second occurrence of 2 does, so the answer is 3.For a = [2, 4, 3, 5, 1], the output should be firstDuplicate(a) = -1. <pre>numbers=[1,2,3,6,0,3,2] def firstDuplicated(numbers): numberWithotRep=list(set(numbers)) duplicated=[] duplicatedDict={i:0 for i in numberWithotRep} #(rakam,tekrarlan index) for i,k in enumerate(numberWithotRep): for j,k in enumerate(numbers): if numberWithotRep[i]==numbers[j] :#duplicated duplicatedDict[numberWithotRep[i]]+=1 if duplicatedDict[numberWithotRep[i]]>1 and numberWithotRep[i] not in [i[0] for i in duplicated]: duplicated.append((numberWithotRep[i],j)) if(duplicated): minIndex=duplicated[0][1] for i in duplicated: if(i[1]<minIndex): minIndex=i[1] return numbers[minIndex] return -1 print(firstDuplicated(numbers))</pre> <p>3</p>	
Metodu farklı listeler kullanarak test ettim	
Kontrol Eden:	Onay

Yaprak No: 11	Kısım:
Tarih: 02/07/2018	Yapılan İş: PROJE ANALİZİ
<p>Python Flask Framework ile bir web uygulaması geliştirilecek.</p> <p>Uygulama genel olarak facebook çalışma mantığına çok yakın, her kullanıcının bir hesabı var ve home sayfasında bir gönderi paylaşabiliyor kendi gönderileri düzeltebiliyor ve silebiliyor. Tüm gönderiler home sayfasında diğer kullanıcılardan erişilebiliyor.</p> <p>Uygulamanın temel kısımları:</p> <ol style="list-style-type: none">1- Kullanıcı kayıt sistemi (registration system)2- Kullanıcı giriş sistemi (login system)3- Gönderi paylaşım sistemi <p>Öğrenilmesi gereken teknolojiler</p> <ol style="list-style-type: none">1- temel html2- jinja script dili3- bootstrap kullanımı4- python flask kurulumu ve kullanımı5- veri tabanının (SQLAlchemy) ana işlemleri (veri ekleme ,siline , sorgulama, düzeltme) <div><div><div>Flask Blog Home About Login Register</div><div><div>Corey Schaefer April 18, 2018</div><div>Blog Post 1</div><div>First post content</div></div><div><div>Our Sidebar</div><div>You can put any information here you'd like:</div><div>Latest Posts</div><div>Announcements</div><div>Calendar</div><div>etc</div></div></div><div>Home sayfası</div><div><div>Flask Blog Home About</div><div><div>Log In</div><div>Email</div><div>Password</div><div><input type="checkbox"/> Remember Me</div><div>Login</div><div>Forgot Password?</div></div><div>Need An Account? Sign Up Now</div></div><div>login sayfası</div><div><div>Flask Blog Home About</div><div><div>Join Today</div><div>Username</div><div>Email</div><div>Password</div><div>Confirm Password</div><div>Sign Up</div></div></div><div>logout sayfası</div></div>	
Kontrol Eden:	Onay

Yaprak No: 12	Kısım:
Tarih: 03/07/2018	Yapılan İş: FLASK VE DECORATORLAR
<p>Flask framework hakkında araştırma yapmaya başladım. Flask web uygulamaları yapmamıza yarayan mini bir framework flasktaki kodların çoğu decorator'ler ile başladığı için decorator'ler hakkında araştırma yaptım.</p> <p>Python'daki fonksiyonlar birer obje, bu özellik sayesinde bir fonksiyonu parametre olarak başka bir fonksiyon alabiliyor veya bir fonksiyonunun dönüş değeri de bir fonksiyon olabilir, decorator'lar da, tam bu işi yapmaya yarıyor. Python'da decorator'lar, argüman olarak fonksiyon alıp, sonuçta yine fonksiyon döndüren fonksiyonlardır.</p> <p>Örnek :</p> <pre>def disFonk(f): def icFONK(): return " "+f()+" " return icFONK def f(): return " f donduruldu"</pre> <p>Normal çağırma</p> <pre>a=disFonk(f) a() ' f donduruldu '</pre> <p>decorator'ları kullanarak çağırma</p> <pre>@disFonk def f(): return " f donduruldu" f() ' f donduruldu '</pre> <p>=></p> <p>Sonra flask framework yardımıyla webte basit "hello world" sayfası "cümlesini içeren bir sayfa oluşturdum.</p> <pre>from flask import Flask app=Flask(__name__) @app.route('/') def index(): return " hello word sayfası" if __name__ == '__main__': app.run(debug=True)</pre> <p>hello word sayfası</p> <p>=></p>	
Kontrol Eden:	Onay

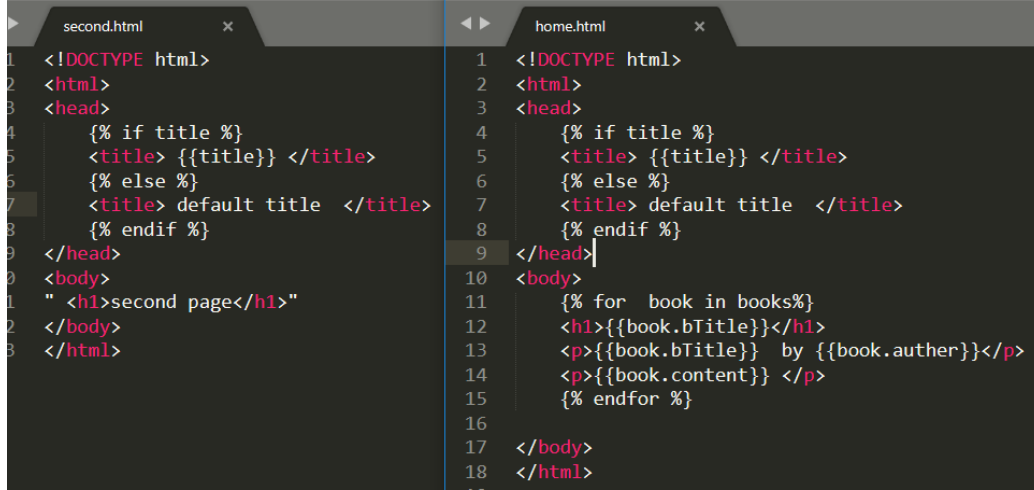
Yaprak No: 13	Kısım:
Tarih: 04/07/2018	Yapılan İş: SUBLİME KURULUMU VE ÖZELLİKLERİ
<p>Bugün hem html hem .py dosyaları oluşturacağım, bunun için sublime text diye bir IDE indirdim bu IDE de birden fazla dili destekliyor, örnek olarak c++,python ,java...</p> <p>Aynı zamanda ileri seviye özelliklere de sahip.</p> <p>IDE' nin Özellikleri</p> <p>1-Multiple Cursor Özelliği</p> <pre>- name: Provision EC2 and RDS infrastructure hosts: localhost connection: local gather facts: False tasks: - name: Create a basic web security group (TCP 22, 80, ec2_group: name: "{{ web_security_group_name }}"</pre> <p>Çoklu seçim, autocomplete ve ona benzer bir sürü özellik destekliyor</p> <p>Proje Oluşturma</p> <p>İdeyi indirdikten sonra bir proje oluşturdum, içinde template diye bir klasör oluşturdum ve içinde iki tane html sayfaları oluşturdum, sonra bu sayfalara render_template metodu kullanarak python projemde erişim sağladım. Dün de aynı örneği yaptım, orada html sayfanın tümü .py dosyanın içindeydi bugün ise kod içinde, sadece html sayfanın ismi yazıldı, o da programın karmaşıklığını azaltıyor</p> <p>Oluşturulan html sayfası</p> <pre><!DOCTYPE html> <html> <head> <title></title> </head> <body> " <h1>main page</h1>" </body> </html></pre> <p>.py dosyasında html sayfasını çağırma</p> <pre>from flask import Flask,render_template app=Flask(__name__) @app.route('/') def home(): return render_template('home.html') @app.route('/second') def second(): return render_template('second.html') if __name__ == '__main__': app.run(debug=True)</pre>	
Kontrol Eden:	Onay

Yaprak No: 14	Kısım:
Tarih: 05/07/2018	Yapılan İş: JINJIA
<p>Geçen örnekte html sayfasındaki metin sabitti, gerçek hayatta dinamik bir yapıya ihtiyaç duyuyoruz. Html sayfasında kod satırları {% %} sembollerin arasına konulur, değişkenler ise {{ }} içindedir, bu jinja diye bir templating dilidir. Örnek olarak Burada şart ifadesi var sayfanın başlığı varsa olduğu gibi bırakılır yoksa default olarak default title adlandırılacak.</p> <pre><!DOCTYPE html> <html> <head> {% if title %} <title> {{title}} </title> {% else %} <title> default title </title> {% endif %} </head> <body> " <h1>main page</h1>" </body> </html></pre> <p>Aynı mantıkla html sayfanın içeriği bilgileri bir listeden okuyarak doldurabilir, bu örnekte kitaplar listesi oluşturuldu, bilgiler listeden html sayfasına aktarıldı.</p> <pre>books=[{ "author":"mark" ,"bTitle":"book1" ,"content":"content of the book1" } , { "author":"ali" ,"bTitle":"book2" ,"content":"content of the book2" }] <body> {% for book in books%} <h1>{{book.bTitle}}</h1> <p>{{book.bTitle}} by {{book.author}}</p> <p>{{book.content}}</p> {% endfor %} </body></pre> <p>Oluşturulan sayfa</p> 	
Kontrol Eden:	Onay

Yaprak No: 15	Kısım:
Tarih: 05/07/2018 (devamı)	Yapılan İş: JINJIA
<p>“Sayfa kaynağını görüntüle” ye bastığımda html sayfasında yazmadığım satırların dinamik bir şekilde üretildiğini görebiliriz.</p>  <pre><!DOCTYPE html> <html> <head> <title> main title </title> </head> <body> <h1>book1</h1> <p>book1 by mark</p> <p>content of the book1 </p> <h1>book2</h1> <p>book2 by ali</p> <p>content of the book2 </p> </body> </html></pre>	
Kontrol Eden:	Onay

Yaprak No: 16**Kısım:****Tarih: 06/07/2018****Yapılan İş: TEMPLATE INHERITANCE**

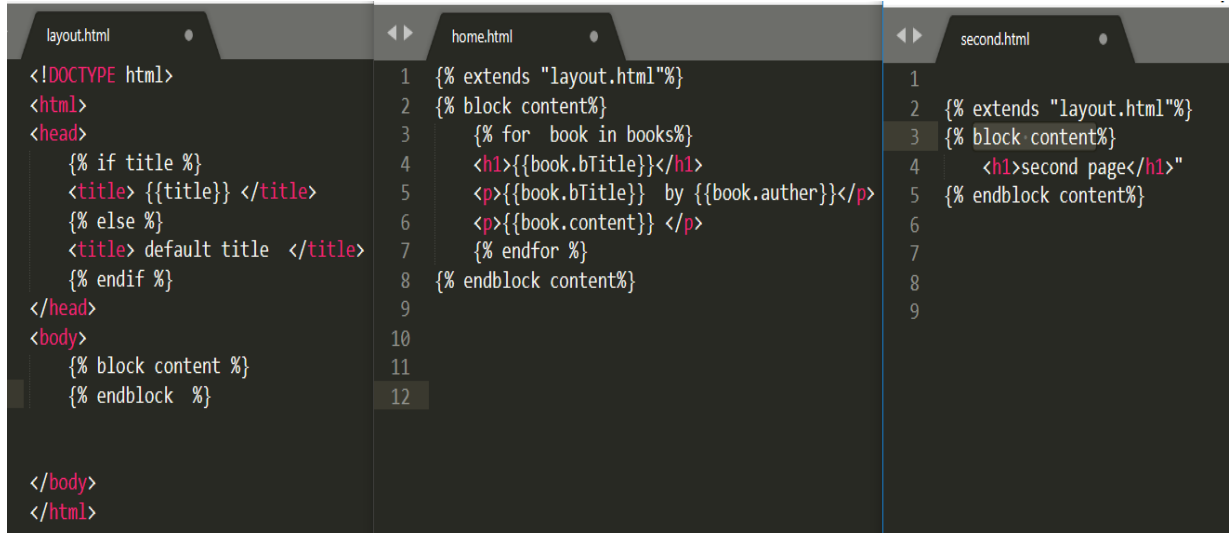
Bugün araştırılacak konu template inheritance, bu koda bakarsak second sayfasında home sayfasındaki kodların çoğu ortak olduğunu görebiliyoruz.



```
second.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   {% if title %}
5   <title> {{title}} </title>
6   {% else %}
7   <title> default title </title>
8   {% endif %}
9 </head>
10 <body>
11   " <h1>second page</h1>"
12 </body>
13 </html>

home.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   {% if title %}
5   <title> {{title}} </title>
6   {% else %}
7   <title> default title </title>
8   {% endif %}
9 </head>
10 <body>
11   {% for book in books%}
12   <h1>{{book.bTitle}}</h1>
13   <p>{{book.bTitle}} by {{book.author}}</p>
14   <p>{{book.content}} </p>
15   {% endfor %}
16
17 </body>
18 </html>
```

Tekrardan kaçınmak için layout diye bir template oluşturdum, sonra second ve home sayfaları o layoutten extend ettim. Sadece body kısmı değişeceği için body kısmındaki block contenti alt sayfalarda override ettim. Topldım



```
layout.html
<!DOCTYPE html>
<html>
<head>
  {% if title %}
  <title> {{title}} </title>
  {% else %}
  <title> default title </title>
  {% endif %}
</head>
<body>
  {% block content %}
  {% endblock %}
</body>
</html>

home.html
1 {% extends "layout.html"%}
2 {% block content%}
3   {% for book in books%}
4   <h1>{{book.bTitle}}</h1>
5   <p>{{book.bTitle}} by {{book.author}}</p>
6   <p>{{book.content}} </p>
7   {% endfor %}
8 {% endblock content%}
9
10
11
12

second.html
1
2 {% extends "layout.html"%}
3 {% block content%}
4   <h1>second page</h1>"
5 {% endblock content%}
6
7
8
9
```

Layout .html

home.html

second.html

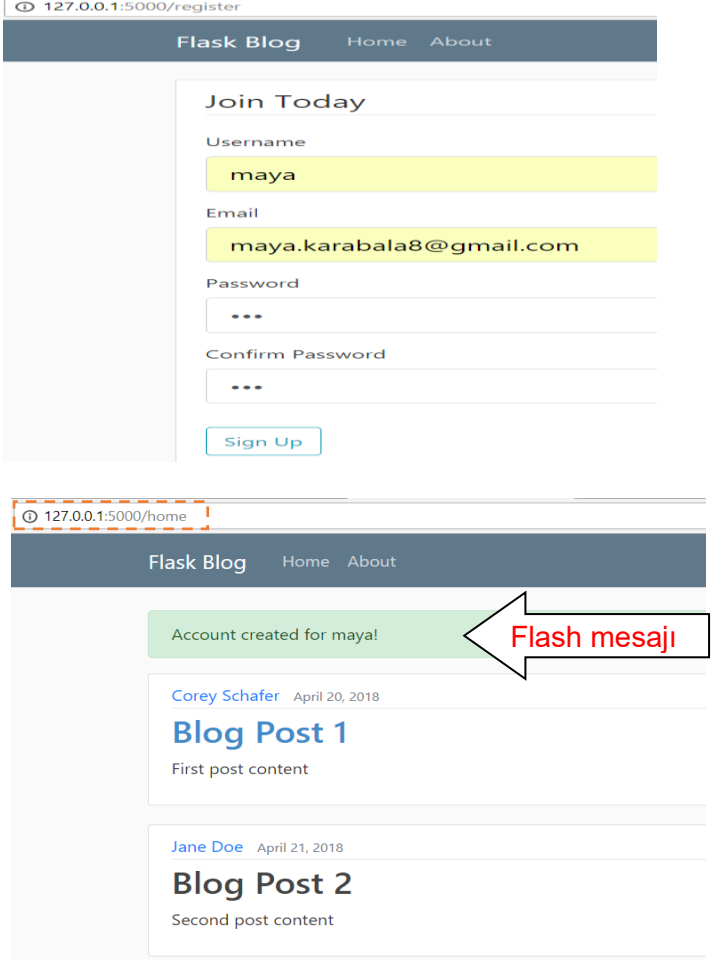
Kontrol Eden:**Onay**

Yaprak No: 17	Kısım:
Tarih:09/07/2018	Yapılan İş: BOOTSTRAP VE VALIDATOR
<p>Bugün Bootstarap hakkında araştırmayla başladım. Bootstarap açık kaynak kodlu, ücretsiz bir CSS frameworktür, tasarım aracıdır. Yüzlerce kod satırın yerine metodları çağırarak işi kolay ve güzel bir tasarımla gerçekleştirilebilir. Desteklediği başka bir özellik “mobile friendly” yani tasarımlar, hem bilgisayarda hem telefon ekranlarında düzenli bir şekilde erişiliyor. Bootstarap frameworku kullanarak sayfanın tasarımı değiştirdim. Books listesine benzer, posts diye bir liste oluşturdum, bilgiler listeden okuyarak hml sayfasında yazdırdım.</p> <div><div><p>Eski hali</p></div><div><p>bootstrsp kullandıktan sonra</p></div></div> <p>Form bilgileri kullanıcıdan aldığımızda yaptığımız genel kontroller değişmiyor, örnek olarak ad alanı boş bırakılmaz, rakam içermmez ve ona benzer akla gelen bütün kontroller wtforms.validators modülünde implement edildi :</p> <p>Validators örnekleri:</p> <ul style="list-style-type: none">DataRequired= bu alan boş bırakılmazLength(a,b)= iki int argüman alıyor bu alan belirlenen sınırları aşamaz ,Email = mailin geçerliğini kontrol ediyor,EqualTo(verilen)= eşitliği kontrol ediyor <pre>from flask_wtf import FlaskForm from wtforms import StringField,PasswordField from wtforms.validators import DataRequired,length,Email,EqualTo class RegistrationForm(FlaskForm): username=StringField("username", validators=[DataRequired(),length(min=2,max=20)]) email=StringField("Email",validators=[DataRequired(),email()]) Password=PasswordField("Password",validators=[DataRequired()]) confirm_password=PasswordField("confirm password",validators=[DataRequired(),EqualTo(password)])</pre>	
Kontrol Eden:	Onay

Yaprak No:18	Kısım:
Tarih:10/07/2018	Yapılan İş: HTML ve BOOTSTRAP
<p>Bugün html hakkında kısa bir araştırma ile başladım.</p> <p>İşaretleme dili olan Html, web sayfalarının hazırlanmasında kullanılan sistemdir. Bir programlama dili olmayan Html bilgisayarlarımızda kullandığımız web sitelerinin oluşturulmasında kullanılır.</p> <p>Örnek Html ve Bootstrap Kodları</p> <ul style="list-style-type: none">- <div> etiketi HTML belgesinde bir bölüm, sektör tanımlar.- <legend> etiketi bir başlık tanımlar.- Resimdeki kodlarla HTML sayfasına mobil friendly özelliği ekleyebiliyoruz. <pre><meta charset="utf-8"> <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"></pre> <ul style="list-style-type: none">- HTML sayfası başka bir css uzantılı sayfa ile bağlanması1. Link tagında sayfayı direk online bootstrap sitesinden alınana bir css sayfaya bağlıyor.2. Link tagında bir özellik, bizim oluşturduğumuz css sayfasında override edilmişse override edilmiş hali html sayfasına uygular. <pre><link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aokXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous"> <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css') }}"></pre> <p>Uygulamada login ve register sayfaları olacak. Bunun için form oluşturmamız lazım. Formu oluşturmak için yapılan adımlar :</p> <ol style="list-style-type: none">1- Forms diye .py uzantılı bir sayfa oluşturdum.2- RegistrationForm ve loginForm classları resimdeki gibi implement edildi.3- Bilgileri kontrol etmek için validotarları ekledim. <pre>forms.py from flask_wtf import FlaskForm from wtforms import StringField,PasswordField,SubmitField,BooleanField from wtforms.validators import DataRequired,length,Email,EqualTo class RegistrationForm(FlaskForm): username=StringField("username", validators=[DataRequired(),length(min=2,max=26)]) email=StringField("Email",validators=[DataRequired(),Email()]) Password=PasswordField("Password",validators=[DataRequired()]) confirm_password=PasswordField("confirm password",validators=[DataRequired(),EqualTo(Password)]) submit=StringField("sign up") class loginForm(FlaskForm): email=StringField("Email",validators=[DataRequired(),Email()]) Password=PasswordField("Password",validators=[DataRequired()]) remember=BooleanField("remember me") submit=StringField("sign in")</pre>	
Kontrol Eden:	Onay

Yaprak No:19	Kısım:
Tarih:10/07/2018 (devamı)	Yapılan İş: FORM İŞLEMLERİ
<p>4 –Main sayfasında loginForm ve RegistrationForm sınıflarından birer obje oluşturuldu ve render_template metoduna parametre olarak girildi.</p> <pre>1 @app.route("/login") 2 def login(): 3 form=loginForm() 4 return render_template('login.html', title='login',form=form) 5 6 @app.route("/register") 7 def register(): 8 form=RegistrationForm() 9 return render_template('register.html', title='register',form=form)</pre> <p>5- Programın güvenliği artırmak için app'ın secret keysini set ettim.</p> <pre>app.config["SECRET_KEY"]="6004df5333dd4eedfde53a3a82454592"</pre> <p>6- Register .html sayfasını oluşturdum ve form bileşenleri html sayfasında gözükebilmesi için gereken kodları ekledim.</p> <p>4.Satırda method tipi GET ya da POST olabilir. Formdan bilgileri göndermek istiyorsak method tipi POST olur.</p> <p>Bir sayfadan bilgi almadan sadece o sayfaya erişmek istediğimizde method tipi GET olur.</p> <pre>1 {% extends "layout.html" %} 2 {% block content %} 3 <div class="content-section"><!--css sayfasından tasarımı alıyor --> 4 <form method="POST" action=""> 5 {{form.hidden_tag()}}<!-- protect our form from certain attack--> 6 <fieldset class="form-group"> 7 <legend "border-bottom mb-4"> Join today</legend> 8 <div class="form-group"> 9 {{ form.username.label (class="form-control-label")}}<!--label kısmı--> 10 {{ form.username (class="form-control form-control-lg")}}<!--textfield kısmı--> 11 </div> 12 </fieldset> 13 </form> 14 </div> 15 {% endblock content %}</pre> <p>Oluşturulan register sayfasının son hali</p> <div><div>127.0.0.1:5000/register</div><div>Flask Blog</div><div>Join today</div><div>username <input type="text"/></div></div>	
Kontrol Eden:	Onay

Yaprak No:20	Kısım:
Tarih: 11/07/2018	Yapılan İş: REGISTER FORMU
<p>Register Formun Devamı</p> <p>7- Altıncı adımda yapılan işlemleri tekrarlayarak formu tamamladım.</p>  <p>Formun altında küçük yazı small tagı kullanarak ekledim yanındaki “sign in” ‘e bastığımız zaman login sayfasını açılır.</p>  <pre><small class="text-muted"> already have an account sign in </small></pre> <p>Formdaki “Sign up” butonuna bastığımız zaman hata alıyoruz Sebebi de kullanıcının girdiği bilgilerin post edilmemesi, bunu çözmek için Test.py sayfasındaki register routuna methods listesini ekledim.</p> <p>Kullanıcı, bilgileri doğru girerse “account created” mesajı flash metodu kullanarak yazdırdım.</p> <p>Flash iki argüman alıyor, Birincisi mesaj metni, ikincisi mesaj türü sonra redirect metodu ile kullanıcıyı home sayfasına yönlendirdim.</p> <pre>@app.route("/register", methods=["GET", "POST"]) def register(): form=RegistrationForm() if form.validate_on_submit(): flash(f" Account created for {{form.username.data}}", "success") return redirect(url_for("home")) return render_template('register.html', title='register', form=form)</pre>	
Kontrol Eden:	Onay

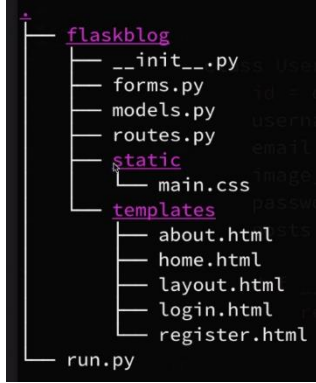
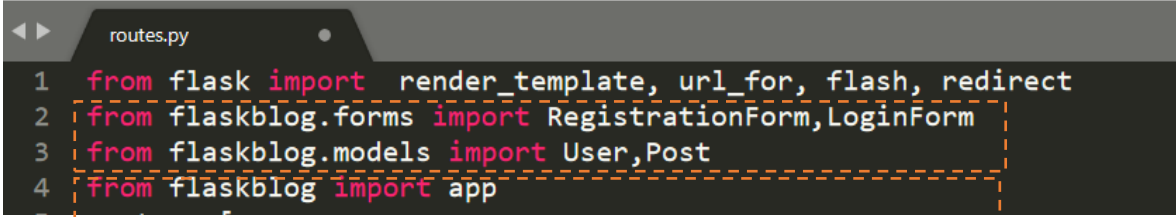
Yaprak No:21	Kısım:
Tarih: 11/07/2018 (devamı)	Yapılan İş: REGISTER FORMU
<p>Flash mesajları bütün sayfalarda göstermek için layout sayfasında blok content üzerinde get_flashed_messages kullanarak mesajlar varsa for ile gezer ve hepsini sayfanın başında gösterir.</p> <pre>{% with messages = get_flashed_messages(with_categories=true) %} {% if messages%} {% for category, message in messages%} <div class="alert alert-{{ category }}"> {{message}} </div> {% endfor %} {% endif %} {% endwith %} {% block content %}{% endblock %}</pre> <p>Bilgileri doğru girdiğimizde home sayfasına yönlendirir ve flash mesajları gösterir.</p> 	
Kontrol Eden:	Onay

Yaprak No: 22	Kısım:
Tarih: 13/07/2018	Yapılan İş: COOKIES VE HATALAR
<p>Cookies hakkında araştırma yaptım.</p> <p>Cookies (Çerez) girmiş olduğumuz web sitesinde o web sitesinin bir daha siteyi ziyaret ettiğimizde bizi tanınması için harddisk yerleştirilen bir text dosyasıdır. Bu text dosyası sadece girdiğimiz o site tarafından okunabilir böylelikle bizi tanınması mümkün olur.</p> <p>Cookieslerin olumlu ve olumsuz yanları var örneğin;</p> <p>Siteler çerezlerle şunları yapabilir:</p> <p>Oturumumuzu açık tutabilir tercihlerimizi hatırlayabilir</p> <p>Alışveriş sepetinde aldığımız şeyleri text dosyasına yazabilir ve bize uygun reklamlar çıkarır.</p> <p>Diğer yandan bazı siteleri bilgileri taplayıp satar ya da kötü bir şekilde kullanır.</p> <p>Register formun devamı</p> <p>Kullanıcı bilgileri hatalı girdiği halde aynı sayfada kalacak ama hata türünden kullanıcıya bilgi vermeyecek. Bunu çözmek için her input faield için resimdeki kodları ayrı ekledim.</p> <p>Labeli yazdırdıktan sonra error varsa textfildin altında hataları yazdırır yoksa text field normal şekilde gösterir.</p> <div><pre><div class="form-group"> {{ form.email.label(class="form-control-label") }} {% if form.email.errors %} {{ form.email(class="form-control form-control-lg is-invalid") }} <div class="invalid-feedback"> {% for error in form.email.errors %} {{ error }} {% endfor %} </div> {% else %} {{ form.email(class="form-control form-control-lg") }} {% endif %} </div></pre></div> <div><p>Username</p><input type="text" value="a"/><p>Field must be between 2 and 20 characters long.</p><p>Email</p><input type="text" value="111111"/><p>Invalid email address.</p><p>Password</p><input type="password"/><p>Confirm Password</p><input type="password"/><p>Field must be equal to password.</p></div> <p>Register.html</p> <p>Hataları gösterim şekli</p>	
Kontrol Eden:	Onay

Yaprak No: 23	Kısım:
Tarih: 16/07/2018	Yapılan İş: SQLAlchemy
<p>Bugün oluşturduğumuz siteye bilgileri temelli saklamak için veri tabanı ekileceğiz. Pythonun desteklediği ve OOP mantığıyla çalışan SQLAlchemy kullanılacak SQLAlchemy modülü terminalden pip komutu kullanarak indirdip test sayfasında import ettim. SQLALCHEMY_DATABASE_URI set ederek database'in konumu bilelerdim, sonra SQLAlchemy sınıfından db diye bir obje oluşturdum.</p> <pre>app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db' db = SQLAlchemy(app)</pre> <p>DatabasedeTablo Oluşturma</p> <p>databasede user tablosu oluşturmak için User sınıfı db.model sınıfından türetmemiz gerekiyor. user sınıfın içerisinde değişken adı databasedeki sütünün adı karşılaşıyor. Bu değişken db.Coulumnden türetmemiz lazım.</p> <p>Parameterler:</p> <ul style="list-style-type: none">• birinci parameter datanın tipi olmalı.(String , integer,text ,date ..)• Primary_key: bu data tabloda tekrarlanmaz ve otomatik atanır.• Nullable: true ise bu alan boş kalabilir False ise doldurulması gerekiyor.• Unique: primary key gibi tekrarlanmaz, otomatik atanmaz.• Posts db.relationship sınıfından türemiş. Bu class paramtere olarak hangi classtan türetilmişse, classın ismi alıyor. <p>Not: burda user sınıfına post diye bir column eklemedik sadece tablolar arasında bir ilişki kuruldu.</p> <ul style="list-style-type: none">• Backreef: post sınıfından User sınıfına author keywordla erişmeyi sağlar başka bir deyişle Post sınıfına author columu eklenmiş gibi davranır.• __repr__ metodu override ettiğimiz için user ya da post türünden bir obje print ile yazdırırsak objenin adresi yerinde bilgilerini yazdırır. <pre>class User(db.Model): id=db.Column(db.Integer,primary_key=True) username=db.Column(db.String(20), unique=True,nullable=False) email=db.Column(db.String(20), unique=True,nullable=False) image_file=db.Column(db.String(20),nullable=False,default="default.jpg") password=db.Column(db.String(60),nullable=False) posts =db.relationship("Post",backref="author",lazy=True) def __repr__(self): return f"User('{self.username}','{self.email}','{self.image_file}')</pre> <p>User sınıfı</p>	
Kontrol Eden:	Onay

Yaprak No: 24	Kısım:
Tarih: 16/07/2018(devamı)	Yapılan İş: SQLAlchemy
<ul style="list-style-type: none">ForeignKey: Sütunun datası başka bir sınıfın primary key olduğunu gösteriyor. <pre>class Post(db.Model): id=db.Column(db.Integer,primary_key=True) title=db.Column(db.String(100), nullable=False) date_posted=db.Column(db.DateTime,nullable=False ,default=datetime.utcnow) content=db.Column(db.Text, nullable=False) user_id=db.Column(db.Integer,ForeignKey=("user.id"), nullable=False) def __repr__(self): return f"Post('{self.title}','{self.date_posted}')</pre> <p>Post sınıfı</p> <p>Tabloları oluşturmak için create all komutu kullanılmalı.</p> <p>İki tane user oluşturup database ekledim ve commit komutu yardımıyla önceki komutları onaylayıp çalıştırdım.</p> <pre>db.create_all() user1=User(username="maya",email="m@gmail.com",password="password") db.session.add(user1) user2=User(username="sara",email="s@gmail.com",password="password2") db.session.add(user2) db.session.commit()</pre>	
Kontrol Eden:	Onay

Yaprak No: 25	Kısım:
Tarih: 17/07/2018	Yapılan İş: PACKAGE STRUCTURE
<p>Test.py sayfasında import ifadeleri, flask türünden app objesi oluşturma kodları, User ve Post classları ve html sayfalarına ulaşmak için route ifadeleri bulunmaktadır.</p> <p>Programın karmaşıklığını azaltmak için bu test modülü package structure kullanarak böleceğiz. Models isimli .py uzantılı bir dosya oluşturduk user ve post sınıfları test dosyasından models dosyasına taşıdık, db kullanacağımız için sayfa başında test modülü import ettik.</p> <pre> models.py x from test import db from datetime import datetime class User(db.Model): id = db.Column(db.Integer, primary_key=True) username = db.Column(db.String(20), unique=True, nullable=False) email = db.Column(db.String(120), unique=True, nullable=False) image_file = db.Column(db.String(20), nullable=False, default='default.jpg') password = db.Column(db.String(60), nullable=False) posts = db.relationship('Post', backref='author', lazy=True) def __repr__(self): return f"User('{self.username}', '{self.email}', '{self.image_file}')" class Post(db.Model): id = db.Column(db.Integer, primary_key=True) title = db.Column(db.String(100), nullable=False) date_posted = db.Column(db.DateTime, nullable=False, default=datetime.utcnow) content = db.Column(db.Text, nullable=False) user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False) def __repr__(self): return f"Post('{self.title}', '{self.date_posted}')" </pre> <p>Test dosyasında User ve Post classları kullanmak için models dosyasını import ettim.</p> <pre> test.py 1 2 from flask import Flask, render_template, url_for, flash, redirect 3 from flask_sqlalchemy import SQLAlchemy 4 from forms import RegistrationForm, LoginForm 5 from models import User, Post 6 </pre> <p>Programı çalıştırdığımda hata aldım, sebebi de dairesel(circular) import oluşması. Test dosyasını çalıştırdığımda başta models modülü run etmeyi çalışıyor, models modülü da aynı şekilde run etmeye çalıştığımızda önce test modülü çalışması lazım böylece sonsuz bir döngüye girmiş oluruz.</p> <p>Projeyi daha düzenli hale getirmek için, aynı zamanda da sorunu çözmek için proje klasörü içinde flaskblog diye bir klasör oluşturdum. Klasör içinde __init__.py dosyası oluşturdum, bu dosya içindeki kodlar “ import + klasörün ismi(flaskblog) “ yazdığımızda direk içindeki objelere ulaşabiliyoruz. Bunun için flask obje oluşturma kodlarını bu modüle taşıdım, sonra routes isimli bir modül oluşturdum. Bütün route kodları o modüle taşıdım ve test modülde mainden başka bir kod kalmadığı için ismini testten run.py ‘ ye değiştirdim ve importlar çapraz olmayacak şekilde düzenledim</p>	
Kontrol Eden:	Onay


Yaprak No: 26	Kısım:
Tarih:17/07/2018(devamı)	Yapılan İş: PACKAGE STRUCTURE
<p>Projenin ağacı</p>  <pre>flaskblog ├── __init__.py ├── forms.py ├── models.py ├── routes.py ├── static │ └── main.css ├── templates │ ├── about.html │ ├── home.html │ ├── layout.html │ ├── login.html │ └── register.html └── run.py</pre> <p>Routes modülündeki importlar;</p> <ul style="list-style-type: none">Klasör içindeki modül başka bir modülden import edilecekse, “From + klasör adı.modül adı + Import + istenen objeler” Şekilde import edilir. (2. Ve 3. Satırlardaki import ifaddelerin bir örneğidir)“From + klasör adı + Import + istenen objeler” Bu import ifade şekli sadece __init__ içindeki objelere erişmemizi sağlar.(4.satır)  <pre>1 from flask import render_template, url_for, flash, redirect 2 from flaskblog.forms import RegistrationForm, LoginForm 3 from flaskblog.models import User, Post 4 from flaskblog import app</pre>	
Kontrol Eden:	Onay

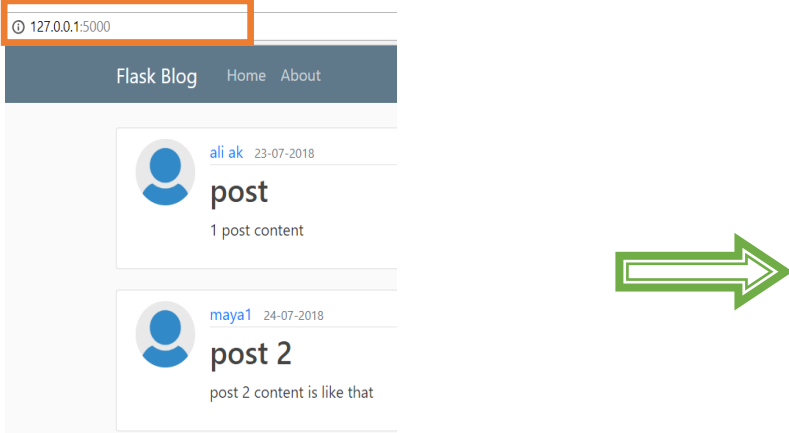
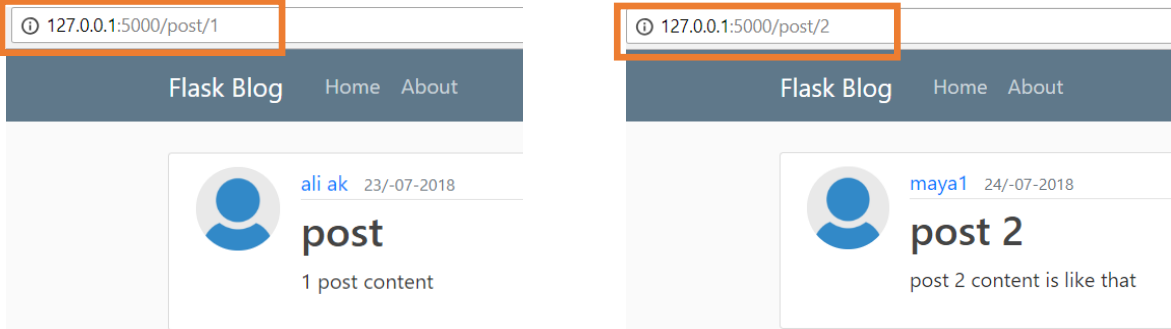
Yaprak No: 27	Kısım:
Tarih:18/07/2018	Yapılan İş: VERİTABANINA KULLANICI EKLEME
<p>Hashing:</p> <p>Register sayfasında girilen bilgilerle yeni bir kullanıcı oluşturup veritabanına eklememiz lazım. Kullanıcının şifresi açık bir şekilde (yani olduğu gibi) veri tabanında tutarsak veritabanına erişen herkes sisteme giriş yapabilecek ve sistem güvensiz olacak. Sistemimizi daha güvenli hale getirmek için şifrelerimizi hashleyerek veritabanında bu hashleri tutmalıyız. Sisteme giriş yapılırken şifrenin doğru olup olmadığını kontrol etmek için ise kullanıcının girdiği şifreyi de hashleyerek bu hashlerin aynı olup olmadığını bakmalıyız.</p> <p>Kullanıcı Ekleme (Register sistemi):</p> <p>Pythonun desteklediği flask-bcrypt hashing iş bize yapıyor.</p> <p>Bcrypt sınıfın metotlarına erişmek için Bcrypt türünden bir obje oluşturmamız lazım.</p> <p>Generate password_hash metodu kullanarak her şifreye byte türünde farklı bir hash ifadesi üretebiliriz, stringe çevirmek için sonuna decode(utf-8) eklememiz lazım.</p> <p>“ check_password_hash ” işimize yarayacak başka bir metod, iki parametre alıyor, birincisi hashlenen şifre ikincisi plain(normal)şifre ve onları karşılaştırarak boolean döndürüyor.</p> <pre>71 from flask_bcrypt import Bcrypt 72 bcrypt=Bcrypt() 73 hashed_pass_byte=bcrypt.generate_password_hash("testing") 74 print(hashed_pass_byte) 75 hashed_pass_string=bcrypt.generate_password_hash("testing").decode("utf-8") 76 print(hashed_pass_string) 77 print(bcrypt.check_password_hash(hashed_pass_string,"other")) 78 print(bcrypt.check_password_hash(hashed_pass_string,"testing"))</pre> <p>b'\$2b\$12\$xDUjsyN4unVhuXODnrK.Q.BYv91onNT0yN6vFI7xd8sBJKce5ikLe' \$2b\$12\$oQPCFWJD5UuT5C75VpHoFOftFSy0tSd1Fdq7aVftbao3JAbcZZv1K False True</p> <p>Byte türünden String türünden</p> <p>Routes modülüne register metodunda formun bilgileri doğru girilirse şifreye hashleyerek veritabanına yeni bir user ekledim ve kullanıcıya login sayfasına yönlendirdim.</p> <pre>29 @app.route("/register", methods=['GET', 'POST']) 30 def register(): 31 form = RegistrationForm() 32 if form.validate_on_submit(): 33 hashed_password=bcrypt.generate_password_hash(form.password.data).decode("utf-8") 34 user=User(username=form.username.data,email=form.email.data,password=hashed_password) 35 db.session.add(user) 36 db.session.commit() 37 flash(f'your account has been created you are now able to log in', 'success')#success bootstrap class 38 return redirect(url_for('login')) 39 return render_template("register.html", title="Register", form=form)</pre>	
Kontrol Eden:	Onay

Yaprak No: 28	Kısım:
Tarih:19/07/2018	Yapılan İş: ÖZEL VALIDATOR EKLEME VE LOGIN SİSTEMİ
<p>Geçen gün yazdığımız kodlarla kullanıcıdan alınan bilgilerle yeni user oluşturabiliyorduk ve veri tabanına ekliyorduk, ama aynı kullanıcı ismiyle ya da aynı maille yeni bir kullanıcı oluşturmaya çalışsak databaseden kaynaklanan bir hata alırız. Sebebi de username ve mail adresinin, unique olmasıdır. Bu yüzden bu kontroller oluşturduğumuz RegistrationForm validatorlere ekleyememiz gerekiyor.</p> <p>Validate_username diye bir metod oluşturdum parametre olarak aldığı username'i kullanarak veritabanında sorgulama yapıyor. Eğer user null değilse yani aynı username sahibi olan bir user bulunduyorsa hata mesajı username textfieldin altında gösterilecek.</p> <ul style="list-style-type: none">• Not: Aynı işlemler mail için yapıldı.• User.query.filter by(şart) metodu şart sağlayan tüm objeleri döndürüyor. <div><pre>def validate_username(self,username): user=User.query.filter_by(username=username.data).first() if user: raise ValidationError("That user name is taken. please choose another") def validate_email(self,email): user=User.query.filter_by(email=email.data).first() if user: raise ValidationError("That user email is taken. please choose another")</pre><div><div>Username</div><div><input type="text" value="maya"/></div><div>That user name is taken. please choose another</div></div><div><div>Email</div><div><input type="text" value="maya.karabala8@gmail.com"/></div><div>That user email is taken. please choose another</div></div></div> <p>Login sistemi</p> <p>Login routunda kullanıcının girdiği mail adresini alıp veritabanında var olup olmadığını kontrol ediyorum, varsa şifre kontrolü yapılıyor ve şifre doğruysa giriş tamamlanıyor. Kullanıcı home sayfasına yönlendirir, bu iki kontrolden biri yanlış gelirse hata mesajı gelir.</p> <pre>42 @app.route("/login", methods=['GET', 'POST']) 43 def login(): 44 form = LoginForm() 45 if form.validate_on_submit(): 46 user=User.query.filter_by(email=form.email.data).first() 47 if user and bcrypt.check_password_hash(user.password,form.password.data): 48 login_user(user,remember=form.remember.data) 49 flash('You have been logged in!', 'success') 50 return redirect(url_for('home')) 51 else: 52 flash('Login Unsuccessful. Please check email and password', 'danger') 53 return render_template('login.html', title='Login', form=form)</pre> <p>userMixin sınıfı kullanıcı durumu hakkında bize bilgi verir, verdiği en önemli özellikler;</p> <p>is_authenticated :giriş yapmış mı (logged in), is_active :aktif mi ,is_anonymous : bilinmeyen bir kullanıcı mı. Bu bilgilere ulaşmak için user sınıfı db.model yanında UserMixin sınıfından türettik.</p> <pre>class User(db.Model,UserMixin):</pre>	
Kontrol Eden:	Onay

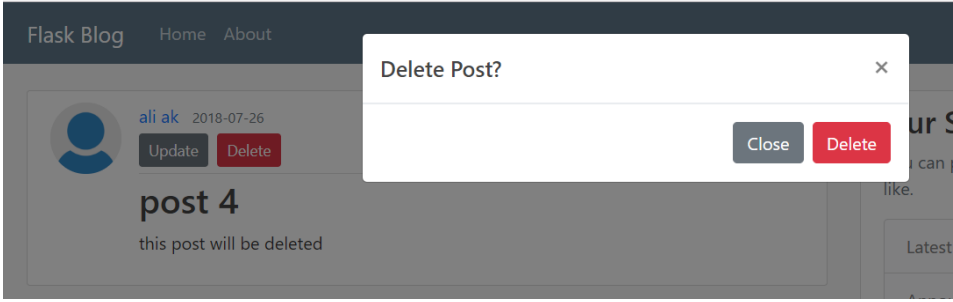
Yaprak No: 29	Kısım:
Tarih: 20/07/2018	Yapılan İş: LOGIN SİSTEMİ
<p>Kullanıcı giriş yaptıktan sonra, sayfanın başında login ve registre'in yerin logout ve account bağlantılarını istiyoruz.Bunun için layout sayfasında bazı değişiklikler yapmamız lazım.</p>  <p>Kullanıcı giriş yapmışsa logout ve account bağlantılarını Göster, yapmamışsa login ve registeri göster.</p> <div><pre><div class="navbar-nav"> {% if current_user.is_authenticated %} logout account {% else %} Login Register {% endif %} </div></pre></div> <div><pre>@app.route("/logout") def logout(): logout_user() return redirect(url_for('home')) @app.route("/account") def account(): return render_template('account.html')</pre></div> <p>Layout.html sayfası</p> <p>routes.py sayfası</p> <p>Bazı sayfalar güvenlik açısından kullanıcı giriş yapmamışsa açılmaması lazım ve o sayfa yerine login sayfasını açılmalı, bunun için loginManager'in login sayfasının bağlantısını init dosyasında set etmemiz lazım.</p> <pre>login_manager.login_view="login"</pre> <p>Ve account route fonksiyonuna</p> <pre>@app.route("/account") @login_required</pre> <p>Loginrequired decoratorı eklememiz lazım</p> <p>Bu durumda kullanıcı giriş yaptıktan sonra istediği önceki sayfaya yönlendirmemiz lazım.</p> <p>Başta flaskten request objesi import etmemiz lazım. Sonra requestin args diye bir dictionary'si var, o dictionary'nin next diye bir özelliği var, bu özellik önceki sayfa varsa bağlantısını, yoksa null döndürür, ona göre kullanıcı yönlendirebiliriz.</p> <pre>next_page=request.args.get("next") return redirect(next_page) if next_page else redirect(url_for('home'))</pre>	
Kontrol Eden:	Onay

Yaprak No: 30	Kısım:
Tarih: 23/07/2018	Yapılan İş: USER ACCOUNT SAYFASI
<p>Account sayfanını başlığı şekildeki gibi kullanıcı bir resim seçmemişse default resmi seçilir ve resmin yanında kullanıcı ismi ve maili gösterilir. Html taglarında <div class="media"> içinde resmin bağlantısı olmalı ve <div class="media-body"> içinde resmin yanındaki metin olmalı.</p> <div><pre>{% extends "layout.html" %} {% block content %} <div class="content-section"> <div class="media"> <div class="media-body"> <h2 class="account-heading">{{current_user.username}}</h2> <p class="text-secondary">{{current_user.email}}</p> </div> </div> <!-- FORM HERE --> </div> {% endblock content %}</pre></div> <div><p>ahmet ahmet@gmail.com</p></div> <p>Account.html sayfası</p> <p>sayfanını görünümü</p> <p>Kullanıcı, account sayfasından bilgileri güncelleyebilir. Bunun için register formuna benzer update_account formu oluşturdum ve gereken eklemeleri account.html sayfasına ekledim. Kullanıcı bazı durumlarda bilgileri değiştirmeden update botuna basabilir bunun için Valdate_username ve Valdate_email de, sadece bilgiler değişmişse tekrar durumu (unique bilgiler)kontrol edilir.</p> <p>Örneğin; sağdaki resimde update işleminde var olan bir kullanıcı ismi ya da maili kullanmaya çalıştığımızda,</p> <div><pre>class UpdateAccountForm(FlaskForm): username = StringField('Username', validators=[DataRequired(), Length(min=2, max=20)]) email = StringField('Email', validators=[DataRequired(), Email()]) submit = SubmitField('Update') def validate_username(self,username): if username.data !=current_user.username: user=User.query.filter_by(username=username.data).first() if user: raise ValidationError("That user name is taken. please choose another") def validate_email(self,email): if username.email !=current_user.email: user=User.query.filter_by(email=email.data).first() if user: raise ValidationError("That user email is taken. please choose another")</pre></div> <div><p>maya2 maya.karabala96@gmail.com</p><p>account info</p><p>Username maya <small>That user name is taken, please choose different one</small></p><p>Email maya.karabala8@gmail.com <small>That email is taken, please choose different one</small></p><p><input type="button" value="Update"/></p></div> <p>Forms.py</p> <p>Routes sayfasında account fonksiyonu bu şekilde implement edildi. Update form bilgileri doğru girilirse bilgileri databasede güncellemek için gereken kodlar (satır 66 ->70).</p> <p>Update formu açıldığında textfieldlerde eski bilgilerin gözükmesi için (satır 72 ->74).</p> <pre>63 @app.route("/account", methods=['GET', 'POST']) 64 @login_required 65 def account(): 66 form=UpdateAccountForm() 67 if form.validate_on_submit(): 68 current_user.username=form.username.data 69 current_user.email=form.email.data 70 db.session.commit() 71 flash("your account has been updated","success") 72 elif request.method=="GET": 73 form.username.data=current_user.username 74 form.email.data=current_user.email 75 image_file=url_for("static",filename="profile_pics/"+current_user.image_file) 76 return render_template('account.html', title='account',image_file=image_file,form=form)</pre>	
Kontrol Eden:	Onay

Yaprak No: 31	Kısım:
Tarih: 24/07/2018	Yapılan İş: GÖNDERİ EKLEME
<p>Gönderi ekleme adımları;</p> <ol style="list-style-type: none">1- Forms.py dosyasında PostForm sınıfı oluşturdum. Sınıfın değişkenleri title (başlık) ve content (içerik) ve submit(gönder butonu). <pre>class PostForm(FlaskForm): title=StringField("Title",validators=[DataRequired()]) content=TextAreaField("Content",validators=[DataRequired()]) submit=SubmitField("Post")</pre> <ol style="list-style-type: none">1. Routes.py dosyasında new_post metodu oluşturdum. Formun bilgileri doğru girilirse post türünden bir nesne oluşturdum başlığı ve içeriği kullanıcıdan aldım. Yazar ise current_user olarak tanımladım ve veri tabanına ekledim. <pre>@app.route("/post/new", methods=['GET', 'POST']) @login_required def new_post(): form=PostForm() if form.validate_on_submit(): post=Post(title=form.title.data,content=form.content.data,author=current_user) db.session.add(post) db.session.commit() flash("your post has been created ","success") return redirect(url_for("home")) return render_template('create_post.html', title="New Post",form=form)</pre> <ol style="list-style-type: none">2. Creat_post.html sayfasını oluşturdum formun bilgileri ve textfioldleri gözükmeleri için gereken kodları yazdım.  <p style="text-align: center;">New post sayfası</p> <ol style="list-style-type: none">3. Home sayfasında oluşturduğum static post dizisinden değerleri okuyup yazdırıyordum, o diziyi sildim ve onun yerine databaseden okunan tüm postları o listeye atadım. <pre>@app.route("/") @app.route("/home") def home(): posts=Post.query.all() return render_template('home.html', posts=posts)</pre>	
Kontrol Eden:	Onay

Yaprak No: 32	Kısım:
Tarih: 24/07/2018 (devamı)	Yapılan İş: GÖNDERİ EKLEME
<p>1. Home sayfasında bir gönderi seçtiğimiz zaman yeni bir sayfa açılacak ve sayfa içinde sadece seçilen gönderi olacak. Bunun için post.html sayfasını oluşturdum ve sayfanın routu resimdeki gibi;</p> <pre>76 @app.route("/post<int:post_id>") 77 def post(post_id): 78 post=Post.query.get_or_404(post_id) 79 return render_template('post.html', title=post.title,post=post)</pre> <p>Önceki sayfalarda route hep statikti. Burada ise < > sembollerini kullanarak route'un ismini dinamik yapabildik.</p> <p>Post sayfasının içeriğinin home sayfasından tek bir farkı var; home sayfasında bütün gönderiler gözükyor burada ise sadece render template metoduna verilen gönderi parametresinin bilgileri.</p> <p>Not: Home sayfasındaki gönderilerin başlıkları birer sayfa bağlantısı.</p>  <p>Home sayfası</p> <p>Home sayfasından ikinci ya da birinci gönderi seçtiğimizde seçilen gönderinin sayfası açılır.</p> 	
Kontrol Eden:	Onay

Yaprak No: 33	Kısım:
Tarih: 25/07/2018	Yapılan İş: GÖNDERİ GÜNCELLEME
<ul style="list-style-type: none"> Gönderi güncelleme routu” /post/post_id/update “ olarak tanımladım. (şekilde satır 82) Güncelleme yapabilmek için kullanıcı, giriş yapmalıdır. (login required)(satır 83) Update_post metodu parametre olarak post_id alıyor. (84) post değişkenine girilen post_id ye göre databaseden gönderi (post) sorgulanır. (85) Get_or_404 databasede istenen gönderi bulunursa post türünden obje döndürür yoksa “not found” Http hatası verir. Her kullanıcı kendi yazdığı postların güncelleme hakkı var. Kullanıcı yazmadığı bir gönderini güncellemeye çalışırsa 403 numaralı forbidden Http hatası alır. (86-87) Formun bilgileri doğru bir şekilde gönderilirse postun bilgileri databasede güncellenir ve kullanıcıyı post sayfasına yönlendirir. (89-92) Update sayfayı ilk açtığımızda seçtiğimiz postun eski bilgilerin gözükecek. (95-97) Update işlemi için önceden kullandığımız create_post sayfasına birkaç şey ekleyerek kullanılacak. 	
<pre> 82 @app.route("/post/<int:post_id>/update", methods=['GET', 'POST']) 83 @login_required 84 def update_post(post_id): 85 post=Post.query.get_or_404(post_id) 86 if current_user!=post.author: 87 abort(403) 88 form=PostForm() 89 if form.validate_on_submit(): 90 post.title=form.title.data 91 post.content=form.content.data 92 db.session.commit() 93 flash("Your post has been updated","success") 94 return redirect(url_for("post",post_id=post.id)) 95 if request.method=="GET": 96 form.title.data=post.title 97 form.content.data=post.content 98 return render_template('create_post.html', title="update post",form=form,Legend="Update Post") </pre>	
<p>kullanıcı kendi gönderisinin sayfasına girmişse update butonu gözükecek</p>	
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre> {% if post.author == current_user %} <div> update </div> {% endif %} </pre> </div> <div style="width: 50%;">   </div> </div>	
<p>Post.html sayfasında update butonun ekleme kodları</p>	
<p>sayfanın görünümü</p>	
Kontrol Eden:	Onay

Yaprak No: 34	Kısım:
Tarih: 26/07/2018	Yapılan İş: GÖNDERİ SİLME
<p>Post sayfasına silme butonu ekledim. Kullanıcı butona bastığında bir pencere açılacak. Kullanıcı onayladığında gönderi databaseden silinecek, bunun için butonu “deleteModal” ile bağladım.</p> <pre><button type="button" class="btn btn-danger btn-sm m-1" data-toggle="modal" data-target="#deleteModal">Delete</button></pre> <p>“deleteModal” penceresini oluşturmak için bootstrap web sitesinde hazır kodlar var. Oran kodları alıp kendi ihtiyaçlarıma göre başlık ve uyarı metni düzenledim.</p> <p>“Delete” butona bastığımız zaman ekran görüntüsü şeklindeki gibi</p>  <p>Burada kullanıcı “Close” butonuna basarsa pencere kapanacak, “Delete” butona basarsa “delete_post” routunu çağırarak.</p> <pre><button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button> <form action="{ url_for('delete_post', post_id=post.id) }" method="POST"> <input class="btn btn-danger" type="submit" value="Delete"></pre> <p>Routes dosyasında delete_post metodu oluşturdum bu metod “update_post” metoduna çok benzer. Kullanıcı giriş yapmışsa ve aynı zamandı kendi yazdığı bir gönderi silmeye çalışıyorsa O gönderiyi temelli bir şekilde “db.session.delete” komutuyla silinebilir.</p> <pre>100 @app.route("/post/<int:post_id>/delete", methods=['POST']) 101 @login_required 102 def delete_post(post_id): 103 post = Post.query.get_or_404(post_id) 104 if post.author != current_user: 105 abort(403) 106 db.session.delete(post) 107 db.session.commit() 108 flash('Your post has been deleted!', 'success') 109 return redirect(url_for('home'))</pre>	
Kontrol Eden:	Onay

Yaprak No: 35	Kısım:
Tarih: 27/07/2018	Yapılan İş: ÖZEL HATA SAYFALARI
<p>Kullanıcı, olmayan bir syafaya ulaşmayı çalıştığında 404 numaralı “Not Found” hatası alır..</p> <div><div>404 Not Found</div><div>127.0.0.1:5000/TEST</div><div>Not Found</div><div>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</div><div>Aynı şekilde kullanıcı, başka bir kullanıcının yazdığı gönderiyi güncellemeye çalıştığında 403 numaralı “Forbidden” hatası alır.</div><div><div>403 Forbidden</div><div>127.0.0.1:5000/post/3/update</div><div>Forbidden</div><div>You don't have the permission to access the requested resource. It is either read-protected or not readable by the server.</div><div>Bu hataların gösterim şeklini düzeltmek amacıyla bu hata sayfaları uygulamamızda override etmemiz lazım.</div><div>Errors isimli bir klasör oluşturdum ve içinde error handler.py diye bir dosya oluşturdum.</div><div>Uyugulamamızda routes dosyasına benzer olan handlers dosyası bu şekilde implement edildi.</div><div><div>handlers.py</div><div><pre>1 from flask import Blueprint,render_template 2 errors=Blueprint("errors",__name__) 3 4 @errors.app_errorhandler(404) 5 def error_404(error): 6 return render_template("errors/404.html"),404 7 8 @errors.app_errorhandler(403) 9 def error_403(error): 10 return render_template("errors/403.html"),403</pre></div></div><div>Her erroer için ayrı bir html sayfası oluşturdum.</div><div><div>403.html</div><div>404.html</div><div>403.htm</div><div>404.html</div><div>__init__ dosyasında “errors” ‘u imprt ettik</div><div><div>16 from flaskblog.errors.handlers import errors 17 app.register_blueprint(errors)</div><div>Error 403 sayfanın son hali:</div><div><div>127.0.0.1:5000/post/2/update</div><div>Flask Blog Home About</div><div>you don't have permission to do that</div><div>please check your account and try again</div></div></div><div>Kontrol Eden:</div><div>Onay</div></div></div></div>	

