Uygulama Projesi: Müşteri360

Gerekli tahmini süre: 45 dakika

Bu uygulama projesi, Django uygulama geliştirme alanında öğrendiğiniz becerileri uygulamalı olarak deneyimlemenizi sağlayacaktır.

Senaryo

Organizasyonunuz farklı platformları birleştiriyor ve müşteri iletişim kayıtlarını merkezi bir yerde depolamak istiyor. Bir yazılım mühendisi olarak, bu Müşteri360 uygulamasını Django kullanarak geliştirmeniz görevi verildi.

Bir iletişim kaydı Kanal, Yön (gelen veya giden) ve Özeti depolar.

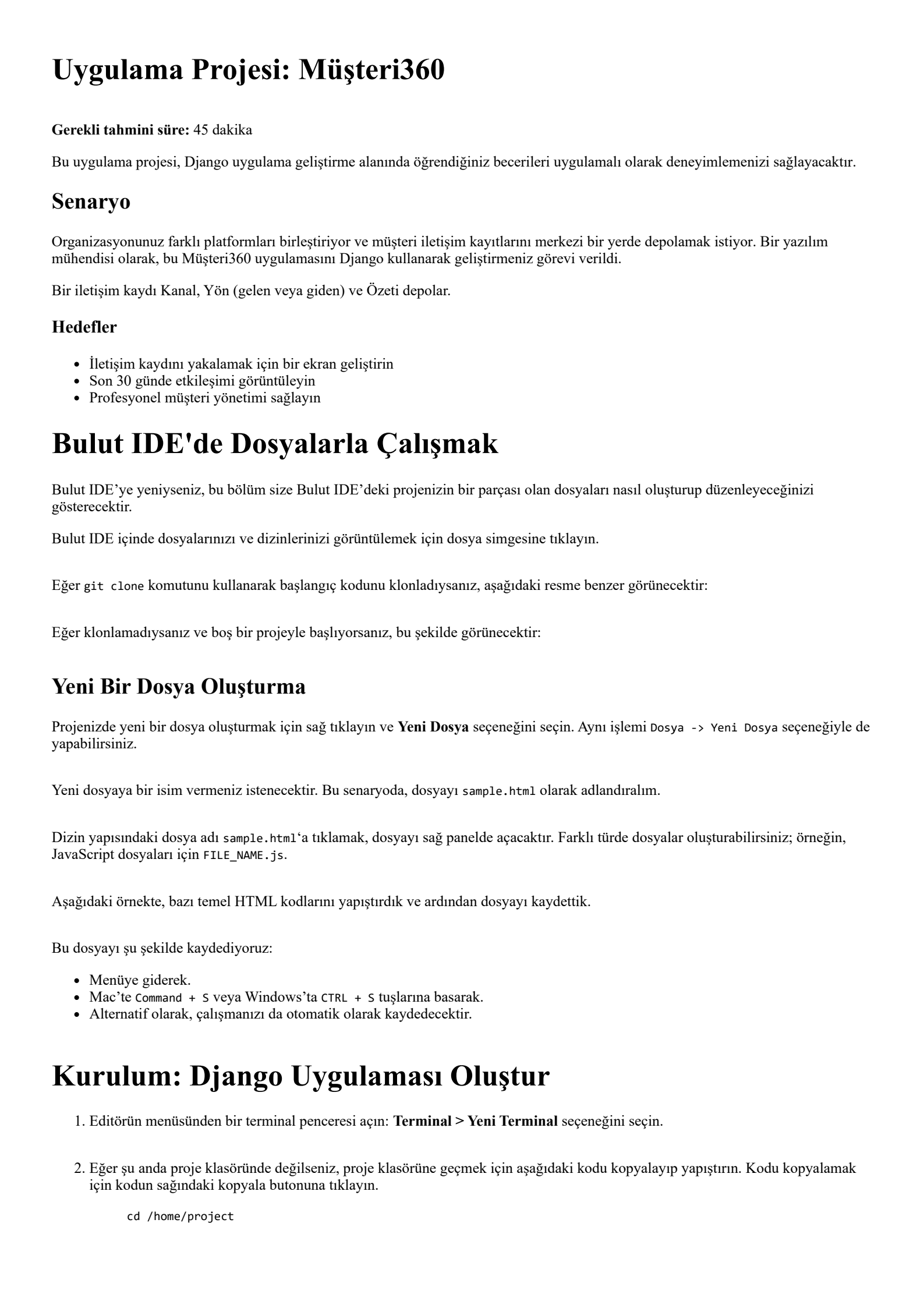
Hedefler

- İletişim kaydını yakalamak için bir ekran geliştirin
- Son 30 günde etkileşimi görüntüleyin
- Profesyonel müşteri yönetimi sağlayın

Bulut IDE'de Dosyalarla Çalışmak

Bulut IDE'ye yeniyseniz, bu bölüm size Bulut IDE'deki projenizin bir parçası olan dosyaları nasıl oluşturup düzenleyeceğinizi gösterecektir.

Bulut IDE içinde dosyalarınızı ve dizinlerinizi görüntülemek için dosya simgesine tıklayın.



Eğer `git clone` komutunu kullanarak başlangıç kodunu klonladıysanız, aşağıdaki resme benzer görünecektir:

Eğer klonlamadıysanız ve boş bir projeyle başlıyorsanız, bu şekilde görünecektir:

Yeni Bir Dosya Oluşturma

Projenizde yeni bir dosya oluşturmak için sağ tıklayın ve **Yeni Dosya** seçeneğini seçin. Aynı işlemi `Dosya -> Yeni Dosya` seçeneğiyle de yapabilirsiniz.

Yeni dosyaya bir isim vermeniz istenecektir. Bu senaryoda, dosyayı `sample.html` olarak adlandıralım.

Dizin yapısındaki dosya adı `sample.html`'a tıklamak, dosyayı sağ panelde açacaktır. Farklı türde dosyalar oluşturabilirsiniz; örneğin, JavaScript dosyaları için `FILE_NAME.js`.

Aşağıdaki örnekte, bazı temel HTML kodlarını yapıştırdık ve ardından dosyayı kaydettik.

Bu dosyayı şu şekilde kaydediyoruz:

- Menüye giderek.
- Mac'te `Command + S` veya Windows'ta `CTRL + S` tuşlarına basarak.
- Alternatif olarak, çalışmanızı da otomatik olarak kaydedecektir.

Kurulum: Django Uygulaması Oluştur

1. Editörün menüsünden bir terminal penceresi açın: **Terminal > Yeni Terminal** seçeneğini seçin.

2. Eğer şu anda proje klasöründe değilseniz, proje klasörüne geçmek için aşağıdaki kodu kopyalayıp yapıştırın. Kodu kopyalamak için kodun sağındaki kopyala butonuna tıklayın.

```
cd /home/project
```

3. pip’in yüklü olduğundan emin olun.

```
python3.11 -m ensurepip
```

4. Django’yu yükleyin.

```
python3.11 -m pip install Django
```

5. Bir proje oluşturun.

```
django-admin startproject customer360
```

6. Laboratuvar ortamında çalışması için dizini değiştirin.

```
cd customer360
```

7. Uygulamayı ilk kez çalıştırmadan önce göç işlemini gerçekleştirin.

```
python3.11 manage.py migrate
```

8. Bu sefer sunucuyu başarıyla çalıştırın.

```
python3.11 manage.py runserver
```

Uygulamayı Başlat

9. Aşağıdaki resim gibi görünecek:

10. Terminalinizde, web sunucunuzu durdurmak için `CTRL+C` tuşlarına basın.

Görev 1: Ayarları Değiştir

Artık ortam projeniz kurulduğuna göre, çalışmaya başlayabilirsiniz.

Django'da yazdığınız her uygulama, belirli bir kurala uyan bir Python paketinden oluşur. Django, bir uygulamanın temel dizin yapısını otomatik olarak oluşturan bir yardımcı program ile birlikte gelir, böylece dizinler oluşturmak yerine kod yazmaya odaklanabilirsiniz.

customer360 uygulamasındaki `settings.py` dosyasında değişiklikler yapacaksınız:

Open **settings.py** in IDE

İzin Verilen Ana Bilgisayarlar

Bu Django sitesinin hizmet verebileceği ana bilgisayar/domain adlarını temsil eden bir dizi dizedir.

```
ALLOWED_HOSTS=["*"]
```

Yüklenen Uygulamalar

Bu Django kurulumunda etkin olan tüm uygulamaları belirten bir dizi. Her dizi, aşağıdakilere işaret eden noktalı bir Python yolu olmalıdır:

- bir uygulama yapılandırma sınıfı (tercih edilir), veya
- bir uygulama içeren bir paket.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'customer360'
]
```

Güvenilir Kaynaklar Listesi

Güvensiz istekler için güvenilir kaynakların bir listesi (örneğin, POST).

Origin başlığını içeren istekler için, Django CSRF koruması başlığın Host başlığında bulunan kaynakla eşleşmesini gerektirir.

```
CSRF_TRUSTED_ORIGINS = ['https://*.cognitiveclass.ai']
```

OS'yi İçe Aktar

`path` özelliğini kullanabilmek için `os` modülünü içe aktarmanız gerekir. Dosyanın üst kısmına, `from pathlib...` ifadesinden sonra bir `import` ifadesi ekleyin.

```
from pathlib import Path
import os
```

Ek Statik Dosya Dizini Yapılandırması

Bu ayar, `FileSystemFinder` bulucusu etkinleştirildiğinde statik dosyalar uygulamasının geçeceği ek konumları tanımlar; örneğin, `collectstatic` veya `findstatic` yönetim komutunu kullanıyorsanız veya statik dosya sunma görünümünü kullanıyorsanız.

Açıklık için, bunu `STATIC_URL`'dan sonra ekleyin.

```
STATICFILES_DIRS = (
    os.path.join(BASE_DIR,"static/"),
)
```

```
"""
    Müşteri360 projesi için Django ayarları.

    'django-admin startproject' ile Django 4.2.4 kullanılarak oluşturuldu.

    Bu dosya hakkında daha fazla bilgi için bakınız
    https://docs.djangoproject.com/en/4.2/topics/settings/

    Ayarların tam listesi ve değerleri için bakınız
    https://docs.djangoproject.com/en/4.2/ref/settings/
```

```
    """
from pathlib import Path
import os
# Proje içindeki yolları şu şekilde oluşturun: BASE_DIR / 'alt_dizin'.
BASE_DIR = Path(__file__).resolve().parent.parent
# Hızlı başlangıç geliştirme ayarları - üretim için uygun değil
# https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
# GÜVENLİK UYARISI: üretimde kullanılan gizli anahtarı gizli tutun!
SECRET_KEY = 'django-insecure-mxj20imb1j!8hz2!kqt*qh5^=y3q3^hyknmj**bpi9v2vuhr!p'
# GÜVENLİK UYARISI: üretimde hata ayıklama açıkken çalıştırmayın!
DEBUG = True
ALLOWED_HOSTS = ["*"]
CSRF_TRUSTED_ORIGINS = ['https://*.cognitiveclass.ai']
# Uygulama tanımı
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'customer360'
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'customer360.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'customer360.wsgi.application'
# Veritabanı
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
# Şifre doğrulama
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Uluslararasılaştırma
# https://docs.djangoproject.com/en/4.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
# Statik dosyalar (CSS, JavaScript, Görseller)
# https://docs.djangoproject.com/en/4.2/howto/static-files/
STATIC_URL = 'static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR,"static/"),
)
# Varsayılan birincil anahtar alanı türü
```

```
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Görev 2: Modelleri Oluştur

Veri depolamamıza ve kullanıcı arayüzünü oluşturmamıza yardımcı olacak modellerimizi tanımlayalım.

> Bir model, verileriniz hakkında tek, kesin bilgi kaynağıdır. Depoladığınız verilerin temel alanlarını ve davranışlarını içerir. Genel olarak, her model tek bir veritabanı tablosuna karşılık gelir.

Modelleri Tanımlayın

`customer360/customer360/models.py` içinde bir model dosyası oluşturarak başlayın. Dosyayı oluşturmak için aşağıdaki scripti çalıştırın.

```
touch /home/project/customer360/customer360/models.py
```

Open **models.py** in IDE

Ve aşağıdakileri tanımlayın:

İthalat

```
from django.db import models
```

Müşteri Modeli

```
class Customer(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    phone = models.CharField(max_length=20)
    address = models.CharField(max_length=200)
    def __str__(self):
        return str(self.id)
```

Etkileşim modeli

```
class Interaction(models.Model):
    CHANNEL_CHOICES = [
        ('phone', 'Phone'),
        ('sms', 'SMS'),
        ('email', 'Email'),
        ('letter', 'Letter'),
    ]
    DIRECTION_CHOICES = [
        ('inbound', 'Inbound'),
        ('outbound', 'Outbound'),
    ]
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    channel = models.CharField(max_length=15, choices=CHANNEL_CHOICES)
    direction = models.CharField(max_length=10, choices=DIRECTION_CHOICES)
    interaction_date = models.DateField(auto_now_add=True)
    summary = models.TextField()
```

Aşağıda tamamlanmış models.py dosyasını görebilirsiniz.

▼ Tamamlanmış models.py

```
from django.db import models
# Modellerinizi burada oluşturun.
class Customer(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    phone = models.CharField(max_length=20)
    address = models.CharField(max_length=200)
    def __str__(self):
        return str(self.id)
class Interaction(models.Model):
    CHANNEL_CHOICES = [
        ('phone', 'Telefon'),
        ('sms', 'SMS'),
        ('email', 'E-posta'),
        ('letter', 'Mektup'),
    ]
    DIRECTION_CHOICES = [
        ('inbound', 'Gelen'),
        ('outbound', 'Giden'),
    ]
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    channel = models.CharField(max_length=10, choices=CHANNEL_CHOICES)
    direction = models.CharField(max_length=10, choices=DIRECTION_CHOICES)
    interaction_date = models.DateField(auto_now_add=True)
    summary = models.TextField()
```

Görev 3: Şablonlar Oluşturma

Artık modellerimizi gösterecek ve kullanıcıların uygulama ile etkileşimde bulunmasını sağlayacak birkaç HTML dosyası eklemeniz gerekiyor.

```
mkdir /home/project/customer360/customer360/templates
touch /home/project/customer360/customer360/templates/add.html
```

```
touch /home/project/customer360/customer360/templates/base.html
touch /home/project/customer360/customer360/templates/index.html
touch /home/project/customer360/customer360/templates/interact.html
touch /home/project/customer360/customer360/templates/summary.html
```

base.html

Modelleri ilgili dosyaya yapıştırırken her HTML içeriğini gözden geçirmelisiniz.

```
{% load static %}
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
        <link rel="stylesheet" href="{% static 'css/main.css' %}">
    </head>
    <body>
        <nav class="navbar navbar-default">
                <ul class="nav navbar-nav">
                    <li>
                        <a style="color:black;" href="/">Home</a>
                    </li>
                    <li>
                        <a  style="color:black;" href="/create">New Customer</a>
                    </li>
                    <li>
                        <a style="color:black;" href="/summary">Summary</a>
                    </li>
                </ul>
        </nav>
        {% block content %}
        {% endblock %}
    </body>
</html>
```

index.html

Sonraki adımda `index.html` dosyasını HTML içeriği ile dolduracaksınız. Bu, bizim açılış sayfamız.

Open **index.html** in IDE

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<html>
    <head>
        <title>Home Page</title>
    </head>
    <script>
       function set_customer(){
        var cinput = document.querySelector('input[name="selected_customers"]:checked');
            if (cinput){
                cid = cinput.value;
                window.location = "/interact/"+cid;
            }
            else
                alert("Please select a customer");
```

```
        }
    </script>
    <body>
        <h1>Welcome to Customer 360</h1>
        <p>Interact and Manage your Customers</p>
        <a class="btn btn-primary" style="font-weight:bold; display:inline" onclick="set_customer()">Interact</a>
        <table class="table">
            <thead>
                <tr>
                    <th>Customer ID</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Phone</th>
                    <th>Address</th>
                    <th>Selected</th>
                </tr>
            </thead>
            <tbody>
                {% for customer in customers %}
                    <tr>
                        <td>{{customer.id }}</td>
                        <td>{{customer.name }}</td>
                        <td>{{customer.email }}</td>
                        <td>{{customer.phone }}</td>
                        <td>{{customer.address }}</td>
                        <td>
                            <input type="radio" name="selected_customers" value="{{ customer.id }}">
                        </td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    </body>
</html>
{% endblock content %}
```

add.html

Yeni bir müşteri eklemek için şablonu oluşturacaksınız. `base.html` dosyasını genişlettiğini göreceksiniz.

Open **add.html** in IDE

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<html>
    <head>
        <title>Add a Customer</title>
    </head>
    <body>
        <h1>Add a new Customer</h1>
        <form class="form" method="post" action="/create/">
            {% csrf_token %}
            <div class="form-group">
                <label for="Name">Name </label>
                <input type="text" name="name" required>
            </div>
            <div class="form-group">
                <label for="Email">Email </label>
                <input type="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="Phone">Phone</label>
                <input type="tel" name="phone" required>
            </div>
            <div class="form-group">
                <label for="Address">Address</label>
                <input type="text" name="address" required>
            </div>
            <button type="submit" class="btn btn-success">Add</button>
            <p> {{ msg }} </p>
        </form>
    </body>
```

```
</html>
{% endblock content %}
```

interact.html

Bir müşteri ile etkileşimi kaydetmek için `interact.html` şablonunu kullanıyoruz.

Open **interact.html** in IDE

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<html>
    <head>
        <title>Interact & Manage</title>
    </head>
    <script>
        function selectButton(element) {
            var buttons = element.parentElement.getElementsByClassName("btn");
            for (var i = 0; i < buttons.length; i++) {
                buttons[i].classList.remove("active");
            }
            element.classList.add("active");
        }
        function check_selected(){
            var dirinput = document.querySelector('input[name="direction"]:checked');
            var chaninput = document.querySelector('input[name="channel"]:checked');
            var summary = document.querySelector('textarea[name="summary"]').value;
            if (!dirinput || !chaninput || summary === ""){
                alert("Please fill all required fields");
                return false;
            }
            return true;
        }
    </script>
    <body>
        <h1>Interact With Your Customers</h1>
        <form class="form" method="post"  onsubmit="return check_selected()" action="#">
            {% csrf_token %}
            <div class="form-group">
                <label>Channel</label>
                <div class="btn-group" data-toggle="buttons">
                    {% for channel in channels %}
                        <label class="btn btn-outline-primary" onclick="selectButton(this)">
                            <input type="radio" name="channel" value="{{ channel.0 }}" required> {{ channel.1 }}
                        </label>
                    {% endfor %}
                </div>
            </div>
            <div class="form-group">
                <label>Direction</label>
                <div class="btn-group" data-toggle="buttons">
                    {% for direction in directions %}
                        <label class="btn btn-outline-primary" onclick="selectButton(this)">
                            <input type="radio" name="direction" value="{{ direction.0 }}" required> {{ direction.1 }}
                        </label>
                    {% endfor %}
                </div>
            </div>
            <div class="form-group">
                <label>Summary</label>
                <textarea name="summary"></textarea>
            </div>
            <button type="submit" class="btn btn-success">Save Interaction</button>
            <p>{{ msg }}<p>
        </form>
    </body>
</html>
{% endblock content %}
```

summary.html

Sonunda, HTML şablonunu `summary.html` aşağıdaki gibi dolduracaksınız:

Open **summary.html** in IDE

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<html>
    <body>
        <h1> Interactions in last 30 Days  </h1>
        {% if not interactions %}
            <p> there are no interactions in the last 30 days <p>
        {% else %}
            <table class="table">
                <thead>
                    <tr>
                        <th>Channel</th>
                        <th>Direction</th>
                        <th>Count</th>
                    </tr>
                </thead>
                <tbody>
                    {% for interaction in interactions %}
                        <tr>
                            <td>{{ interaction.channel }}</td>
                            <td>{{ interaction.direction }}</td>
                            <td> {{ interaction.count }} </td>
                        </tr>
                    {% endfor %}
                </tbody>
            </table>
            <h4> Total : {{ count }} </h4>
        {% endif %}
    </body>
</html>
{% endblock content %}
```

Görev 4: Görünümler Oluştur

Artık `customer360/customer360/views.py` dosyasını oluşturarak görünümleri tanımlayacaksınız.

```
touch /home/project/customer360/customer360/views.py
```

Open **views.py** in IDE

İçe Aktarmaları Tanımla

```
from django.shortcuts import render
from datetime import date, timedelta
from django.db.models import Count
from .models import *
```

İndeks Görünümünü Tanımla

```
def index(request):
    customers = Customer.objects.all()
    context = {"customers":customers}
    return render(request,"index.html",context=context)
```

Müşteri Görünümünü Oluşturma Tanımı

```
def create_customer(request):
    if request.method == "POST":
        name = request.POST["name"]
        email = request.POST["email"]
        phone = request.POST["phone"]
        address = request.POST["address"]
        customer = Customer.objects.create(name=name,email=email,phone=phone,address=address)
        customer.save()
        msg = "Successfully Saved a Customer"
        return render(request,"add.html",context={"msg":msg})
    return render(request,"add.html")
```

Özet Görünümünü Tanımla

```
def summary(request):
    thirty_days_ago = date.today() - timedelta(days=30)
    interactions = Interaction.objects.filter(interaction_date__gte=thirty_days_ago)
    count = len(interactions)
    interactions = interactions.values("channel","direction").annotate(count=Count('channel'))
    context={
                "interactions":interactions,
                "count":count
             }
    return render(request,"summary.html",context=context)
```

Etkileşim Görünümünü Tanımla

```
def interact(request,cid):
    channels = Interaction.CHANNEL_CHOICES
    directions = Interaction.DIRECTION_CHOICES
    context = {"channels":channels,"directions":directions}
    if request.method == "POST":
        customer = Customer.objects.get(id=cid)
        channel = request.POST["channel"]
        direction = request.POST["direction"]
        summary = request.POST["summary"]
        interaction = Interaction.objects.create(
                                    customer=customer,
                                    channel=channel,
                                    direction=direction,
                                    summary=summary)
        interaction.save()
        context["msg"] = "Interaction Success"
        return render(request,"interact.html",context=context)
    return render(request,"interact.html",context=context)
```

Tamamlanan Dosya

Tamamlanan dosyayı aşağıda görebilirsiniz.

▼ Tamamlanan views.py

```
from django.shortcuts import render
from datetime import date, timedelta
from django.db.models import Count
from . models import *
# Görünümlerinizi burada oluşturun.
def index(request):
    customers = Customer.objects.all()
    context = {"customers":customers}
    return render(request,"index.html",context=context)
def create_customer(request):
    if request.method == "POST":
        name = request.POST["name"]
        email = request.POST["email"]
        phone = request.POST["phone"]
        address = request.POST["address"]
        customer = Customer.objects.create(name=name,email=email,phone=phone,address=address)
        customer.save()
        msg = "Müşteri Başarıyla Kaydedildi"
        return render(request,"add.html",context={"msg":msg})
    return render(request,"add.html")
def summary(request):
    thirty_days_ago = date.today() - timedelta(days=30)
    interactions = Interaction.objects.filter(interaction_date__gte=thirty_days_ago)
    count = len(interactions)
    interactions = interactions.values("channel","direction").annotate(count=Count('channel'))
    context={
                "interactions":interactions,
                "count":count
            }
    return render(request,"summary.html",context=context)
def interact(request,cid):
    channels = Interaction.CHANNEL_CHOICES
    directions = Interaction.DIRECTION_CHOICES
    context = {"channels":channels,"directions":directions}
    if request.method == "POST":
        customer = Customer.objects.get(id=cid)
        channel = request.POST["channel"]
        direction = request.POST["direction"]
        summary = request.POST["summary"]
        interaction = Interaction.objects.create(
```

```
                                    customer=customer,
                                    channel=channel,
                                    direction=direction,
                                    summary=summary)
            interaction.save()
            context["msg"] = "Etkileşim Başarılı"
            return render(request,"interact.html",context=context)
        return render(request,"interact.html",context=context)
```

Görev 5: URL'ler Oluşturma

Temiz, şık bir URL şeması, kaliteli bir web uygulamasında önemli bir detaydır. Django, URL'leri istediğiniz gibi tasarlamanıza olanak tanır, herhangi bir çerçeve kısıtlaması olmaksızın.

`urls.py` dosyasında değişiklikler yapacaksınız:

Open **urls.py** in IDE

Görünümleri İçe Aktar

```
from . import views
```

URL Deseni Ekle

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.index,name="index"),
    path('create/',views.create_customer,name='create_customer'),
    path('interact/<int:cid>',views.interact,name='interact'),
    path('summary/',views.summary,name='summary'),
]
```

Aşağıdaki tam dosyayı görebilirsiniz.

▼ Tamamlanmış urls.py

```
from django.contrib import admin
from django.urls import path
from . import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.index,name="index"),
    path('create/',views.create_customer,name='create_customer'),
    path('interact/<int:cid>',views.interact,name='interact'),
    path('summary/',views.summary,name='summary'),
]
```

Görev 6: Stil Ekle

Kullanıcı deneyimini görsel olarak geliştirmek için, uygulamanıza eklemeniz gereken aşağıdaki CSS verilmiştir.

```
mkdir -p /home/project/customer360/static/css
touch /home/project/customer360/static/css/main.css
```

CSS içeriği

```
body{
    text-align: center;
    margin: auto;
}
h1,p{
    font-weight: bold;
}
table{
    margin: auto;
    border: 2px solid #000000;
    margin-top: 50px;
    border-radius: 15px;
    background-color: white;
    text-align: center;
}
th{
    text-align: center;
}
.nav{
    width: 200%;
    border: 2px solid black;
    border-radius: 10px;
}
.nav li{
    margin-right:  6%;
    margin-left:  6%;
    font-size: large;
    font-weight: bold;
}
.form{
    margin: auto;
    margin-top: 20px;
    background-color: white;
    text-align: center;
    width: 50%;
    height: 50%;
    border: 2px solid black;
    border-radius: 10px;
}
.form-group{
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    width:100%;
    padding: 10px;
    text-align: left;
    margin-top: 10px;
    width: 100%;
}
```

```
input[type="text"],input[type="email"],input[type="tel"]{
    width: 100%;
}
textarea{
    width: 500px;
    height: 200px;
}
.btn{
    display: flex;
    margin-top: 10px;
    margin-bottom: 10px;
    padding-left: 50px;
    padding-right: 50px;
    text-align: left;
    margin-left: 10px;
}
.active {
    background-color: #007bff;
    color: #fff;
}
```

Görev 7: Uygulamayı Çalıştır

```
cd /home/project/customer360
python3.11 manage.py makemigrations customer360
python3.11 manage.py migrate
python3.11 manage.py runserver
```

Launch Customer360

Başlat

Customer360 uygulamasını başlattığınızda, aşağıda gösterilen benzer bir görünüm göreceksiniz.

Yeni Müşteri

İlk kez çalıştırdığınız için müşteri kaydı yok. Hadi bir tane ekleyelim.

Müşteri Listesi

Artık ana sayfada yeni eklediğiniz müşteriyi görebilirsiniz.

Yeni Etkileşim kaydı

Bu müşteri için şimdi yeni bir etkileşim kaydedeceksiniz, müşteri kaydını seçip ardından `Etkileşim` seçeneğine tıklayarak.

Sonra etkileşim detaylarını doldurun.

Özet

Artık Özet sayfasına gidebilirsiniz.

Görev 8: Değişiklikler (İsteğe Bağlı)

Şimdiki meydan okumanız, Customer360 uygulamasını geliştirmek ve bazı işlevler eklemektir. Bu değişiklikleri arkadaşlarınızla tartışabilirsiniz.

1. Müşteri modeline yeni bir alan ekleyin
2. Yeni bir etkileşim kanalı oluşturun

Müşteriye sosyal medya Alanı Ekle

Model değişikliği

`Customer` modeline, `social_media` adında yeni bir isteğe bağlı alan ekleyin. Ve bunu müşteri detayları ile birlikte gösterin.

▼ İpucu

```
new_field = models.data_type(args)
```

▼ Çözüm

```
social_media = models.CharField(max_length=100, blank=True)
```

Şablon değişikliği

Artık yeni `social_media` alanının müşteri oluşturma formuna eklenmesini sağlamalısınız.

▼ İpucu

```
<div class="form-group">
    <label for="Field">Alan Etiketi </label>
    <input type="Type" name="Name">
</div>
```

▼ Çözüm

```
<div class="form-group">
    <label for="SocialMedia">Sosyal Medya </label>
    <input type="text" name="social_media">
</div>
```

Ve `social_media` alanının ana sayfadaki gridde eklendiğinden emin olun.

▼ İpucu

```
<th>Alan Adı</th>
...
<td>{{mode.field_name }}</td>
```

▼ Çözüm

```
<th>Sosyal Medya</th>
...
<td>{{customer.social_media }}</td>
```

Görünüm değişikliği

1. Sosyal medya metni olarak gönderim kodunu ekleyin

▼ İpucu

`request.POST` yöntemini kullanın

▼ Çözüm

```
social_media=request.POST["social_media"]
```

2. `Customer.objects.create()` metodunu kullanarak yeni bir Müşteri nesnesi oluşturun.

▼ Ipucu

Mevcut `Customer.objects.create()` metoduna `social_media` niteliğini ekleyin.

▼ Çözüm

```
customer = Customer.objects.create(name=name,email=email,phone=phone,address=address,social_media=social_media)
```

Yeni etkileşim kanalı ekle

Bu değişiklik sadece bir yerde. Bu değişikliğin nerede olması gerektiğini bulabilir misin?

▼ İpucu
Model dosyasında, Kanal Seçenekleri dizisinde yeni bir giriş ekleyin. Bu değişikliğin sadece bir yerde olmasının nedeni, interact.html dosyasında, sabit kodlama yerine olası kanallar üzerinde döngü kurmamızdır.

```
('new_choice', 'Yeni Seçenek'),
```

▼ Çözüm

```
('social_media', 'Sosyal Medya'),
```

Değişikliklerinizi test edin

Bu model değişikliklerini uygulamak için belirli prosedürleri çalıştırmanız gerekiyor. Bu prosedürler, model değişikliklerini veritabanınıza uygulayacaktır.

▼ İpucu

```
# göçler oluştur
# göç et
# testi yapmak için sunucuyu çalıştır
```

▼ Çözüm

```
python3.11 manage.py makemigrations customer360
python3.11 manage.py migrate
python3.11 manage.py runserver
```

Launch Customer360

Özet

Tebrikler!

Django uygulama geliştirme becerilerinizi kullanarak bu pratik projeyi tamamladınız.

Yazar(lar)

- Muhammad Yahya