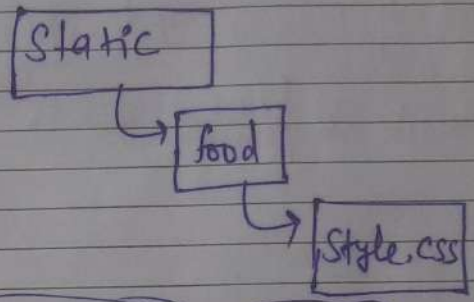
we need to load static or style file here

{% load staticfiles %}

Static files & Site Design

Styling

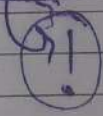
① Static files :-



index.html

in head

<link rel="stylesheet" href="{% static 'food/style.css' %}">



```
<body>
    {% for item in itemList %}
        <ul>
            <li>
                <a>
</body>         </li>
            </ul>
    {% endfor %}
```

<head>
<title> </title>
<link rel="stylesheet" href="{% static 'foo

</head>

<body>

Same

</body>

1 1

d{% style.css %}">

In style.css

```
body {
    background-color: lightyellow;
}
```

Stop server

(-> restart again

/ /

W load static replace with just static as

In recent version → index.html

{% load static %}

then !

More About Static files →

What are Staticfiles ?

Files such as images, JavaScript or CSS.

How does Django know where the static files are located?

/ /

INSTALLED_APPS = [

'django.contrib.staticfiles',

]

STATIC_URL = 'static/'

like - logo

Creating Navbar :-

Navbar → Static

bootstrap → Frontend framework

BootStrapCDN (Content delivery network)

(V4.3)

1 (Bootstrap CDN)

Copy

index.html

↳ in head

↳ below ↳ link:CSS

↳ Paste

How to Create navbar

in body class="navbar" navbar-dark

<nav>

<a href="#" class="navbar-bra

<a class="nav-item" href

nav-link

</nav>

1 1

bg-dark"

d">Food APP

="#">Menu Item

> Add Item <

> Delete Item <

> View Item <

Creating Base Template :-

base.html → {% load static %}

! → in head → bootstrap CDN link : CSS

bootstrap CDN link above title

```
<body>
    <nav>

    </nav>
</body>
```

```
{% block body %}

{% endblock %}
```


In index.html

```
{% extends 'food/base.html' %}
<body>
    {% block body %}

    {% endblock %}
```

In detail.html

```
{% extends 'food/base.html' %}
    {% block body %}

    {% endblock %}
```

Adding Image field to Model

Pizza image → Copy image Address.

In Models.py

item_image = models.CharField(max_length = 500, default = "")

default image

google → Placehold food image

Copy image Address

Python manage.py makemigrations food

Python manage.py sqlmigrate food 002

Python manage.py migrate

bootstrap - 12

Python manage.py runserver

In Index.html

This is old Code

```
{% block body %}
{% for item in item_list %}
<div class="row">
<div class="col-md-3">
</div>
<div class="col-md-4">
</div>
</div>
{% endfor %}
{% endblock %}
```

```
<body>
{% block body %}
{% for item in itemlist %}
<div class = "row">
<div class = "col-md-3 offset-md-2">
<img class = "card" height = "150px" src = "{{ item
</div>

<div class = "col-md-4">
<h3> {{ item.itemname }} </h3>
<h5> {{ item.itemdesc }} </h5>
<h6> ${{ item.itemprice }} </h6>
</div>

<div class = "col-md-2">
<a href = "{
"{% url 'food:detail' item.id %}"
</div>
</div>
```

www.itvedant.com

predict-device

Admin Panel (change image)

```
.itemimage }}" />

{% endfor %}
{% endblock %}
```

ails

www.itvedant.com

Designing the Detail View

In detail.html

```
{% extends 'food/base.html' %}

{% block body %}

<div class = "container" >
  <div class = "row" >
  <div class = "col-md-6">
  <img height = "300px" src = "{{ item.item
  </div>

<div class = "col-md-6" >
<h1> {{ item.item_name }} </h1>
<h2> {{ item.item_desc }} </h2>
<h3> {{ item.item_price }} </h3>
</div>
</div>
</div>
```



```
{% endblock %}
```

-image }}" class = "card" >

Form in Django

Adding form to Add item :-

urls.py

Path setup

```
# add items
path('add', views.create_item, name="creat
```

forms.py

```
from django import forms
from .models import Items

Class ItemForm (forms.ModelForm):
    Class Meta:
        model = Item
        fields = ['item_name', 'item_desc',
                  'item_price', 'item_image']
```


item_form.html

```
<form method="POST">
    {% csrf_token %}  (for security cross side scripting)
    {{form}}
    <button type="submit"> Save </button>

_item"),

</form>
```

views.py

```
from django.shortcuts import
    render, redirect
from .forms import ItemForm

def create_item (request):
    form = ItemForm(request.POST or
                    none)
    if form.is_valid():
        form.save()
        return redirect('food:index')
    return render(request, 'food/item-form.html'
                  , {'form': form})
```

Adding Base Template To Form

No Styling

base.html — localhost:8000/food/add

```
<a class="nav-item nav-link" href=
    "{% url 'food:create_item' %}">
    Add Item </a>
```

now click on add item

item-form.html

```
{% extends 'food/base.html' %}

{% block body %}
<form method="POST">
  {% csrf_token %}
  {{ form }}
  <button type="submit">Save</button>
</form>
{% endblock %}
```


Don't want to add item just Update the existing one :-

Implementing the Edit functionality

(urls.py)

```
#edit
path('update_item/<int:id>/', views.update
    _item, name='update_item'),
```

```
# def update_item(request, id):
    item = item.objects.get(id=id)
    form = ItemForm(request.POST or
                    None, instance
                    =item)
    if form.is_valid():
        form.save()
        return redirect('food:index')

    return render(request, 'food/item-form.html',
        {'form': form, 'item': item})
```

(Same item form for update)

Implementing Delete functionality

#delete

```
path('delete/<int:id>/', views.delete_item,
     name='delete_item'),

def delete_item(request, id):
    item = Item.objects.get(id=id)
    if request.method == 'POST':
        item.delete()
        return redirect('food:index')
    return render(request, 'food/item-delete.
```


detail.html

```
<a href="{% url 'food:delete_item' item.id %}">
    Delete </a>  > Update </a>
<a
```

item-delete.html

```
<form method="POST">
    {% csrf_token %}
    <h2> Are you sure you want to delete
        {{item.item_name}} </h2>
html', {'item': item})
    <button type="submit"> Confirm
</form>
```

Implementing Delete functionality

delete

```
Path('delete/<int:id>/', views.delete_item,
     name='delete_item'),

def delete_item(request, id):
    item = Item.objects.get(id=id)
    if request.method == 'POST':
        item.delete()
        return redirect('food:index')
    return render(request, 'food/item-delete.
```


detail.html

```
<a href="{% url 'food:delete_item' item.id %}">
    Delete </a>
<a                                    >Update </a>
```

item-delete.html

```
<form method="POST">
    {% csrf_token %}
    <h2> Are you sure you want to delete
        {{item.item_name}} </h2>
html', {'item': item})
    <button type="Submit"> Confirm </button>
</form>
```

Authentication in Django

Creating User Registration Form :-

new app → python manage.py startapp users

users → apps.py → UsersConfig

Installed Apps → 'users.apps.UsersConfig',

views.py

Inbuilt forms → from django.contrib.auth.forms import UserCreationForm

```
def register(request):
    form = UserCreationForm()
    return render(request, 'users/register.html',
                  {'form': form})
```


templates → users → register.html

register.html

```
<form method="POST">
  <% csrf_token %>
  {{ form }}
  <button type="submit">Signup</button>
</form>
```

urls.py ~~X~~ instead of → direct on main project

```
from users import views as user_views
path('register/', user_views.register, name='register'),
```

Registeration Success Message

```
from django.shortcuts import redirect, render
from djang.contrib import messages

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            messages.success(request, f"Welcome {username}, your account is created")
            return redirect('food:index')
    else:
        form = UserCreationForm()
    return render(request, 'users/register.html', {'form': form})
```

base.html

below </nav>

```
{% if messages %}
    {% for message in messages %}
        {{message}}
{% endif %}  {% endfor %}
```

Reg

from

from

def

base

below

```
<div class="alert alert-{{message.tags}}">{{message}}</div>
```

~~Saving users~~

```
</nav>
{% if messages %}
  {% for message in messages %}
<div class="alert alert-{{message.tags}}">{{message}}</div>
  {% endfor %}
{% endif %}

{% block body %}
{% endblock %}
```

Refresh Index

WHAT OUR HAPPY LEARNERS SAY..

Adhithi Prabhu

I had enrolled for a **Full Stack Java Development** course and the experience was great. The course was interactive and was provided with assignments and projects for each module. Teaching is very good especially Samruddhi ma'am and Shirish sir as both of them have tremendous knowledge. Would highly recommend joining for Java programming courses.

admin

★★★★★

Users → Same user login → Super user login

Vinod Singh

For months I have been struggling with my career. Thank you Itvedant for your guidance in bringing back my lost dream of being a coder. Itvedant is more than just a hope. With such highly experienced teaching staff and a helpful management, I have no doubt in saying that it is the best IT training institute in India. Thank you for everything.

new user Creation

★★★★★

test user test test 456

test test 456

Amizhthan Murugan

I personally feel the courses are of good quality and I would recommend Itvedant to those who need practical knowledge of technical concepts. Management is strict with attendance, trainers are caring and experienced. Assignments and Assessments are tracked regularly for performance. Great infrastructure with well maintained computers.

★★★★★

www.itvedant.com

/ / is_valid()

Saving Users :- (makes user unique)

Superusername

if form.isvalid():

```
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            messages.success(request, f'welcome {username}
                your account is created successfully')
            return redirect('food:index')
    else:
        form = UserCreationForm()
    return render(request, 'users/register.html',
                  {'form': form})
```

form → extend

1/1

Adding Additional field :-

from .forms import RegisterForm

Users

→ forms.py

User model

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import
                    UserCreationForm

class RegisterForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email',
                  'password1', 'password2']
```

Inherit

newuser1 newuser1@gmail.com

Admin

test test 456

test test 456

1/1

views.py

from .forms import RegisterForm

from django.contrib.auth.forms import
UserCreationForm

def register

Replace UserCreationForm
with RegisterForm

Same Code :-

```
def register(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get
                        ('username')
            messages.success(request,
                f'Welcome {username}, your
                account is created')
            return redirect('food:index')
    else:
        form = RegisterForm()
    return render(request, 'users/register.html',
                  {'form': form})
```

(Login & Log out)

Main project

Logging in Users :-

```
from django.contrib.auth import views

path('login/', authentication_views.loginView
path('logout/', authentication views.
        LogoutView.as_view(),
        name = 'logout'),
```

(Although login & logout are inbuilt still they require template)

```
.as_view (template_name = 'users/login.html')
as_view(template_name = 'users/logout.html')
```

Users → templates → users → login.html, logout.html

Class based Views

```
as authentication views
.as view(), name = 'login'),
```

login.html

```
<form method = "POST">
  {% csrf_token %}
  {{ form }}
  <button type = "submit">login</button>
</form>
```

For log_in → No view is required

In-built view

template required

localhost:8000/login/

registration/login.html

Admin → Log-out

Admin login by SuperUser login

L.h /accounts/profile/
to

settings.py

Log (Below) STATIC_URL = '/static/'

LOGIN_REDIRECT_URL = 'food:index'

Redirecting Registered Users &
Logout Function :-

Admin

SuperUser log-out

localhost:8000/login

login

Username
password

as I hit → login

localhost:8000/accounts/profile/

Profile Page of Users ✗✗ we have not created

/admin — logged-in

Admin → Log-out

Admin login by SuperUser login

Log in /accounts/profile/ to

settings.py

Log (Below) STATIC_URL = '/static/'

LOGIN_REDIRECT_URL = 'food:index'

Redirecting Registered Users & logout function :-

Admin

SuperUser log-out

localhost:8000/login

login

Username

password

as I hit → login

localhost:8000/accounts/profile/

Profile Page of Users ✗✗ we have not Created

/admin — logged-in

Register User → login-page
↳ now login → index

```
def register(request):
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_da
            messages.success(request,
            return redirect('login')

    else:
        form = RegisterForm()
    return render(request, 'users/re
```

fa.get('username')
f'welcome {username}, your account
is Created')

gister.html', {'form': form}

logout.html

```
<h1> You have been logged out
</h1>
```

localhost:8000/logout

log-out

admin

→ login as admin

Two tabs

→ /logout

↓

hL

admin

refresh

logout

Adding login option to navbar

Base.html

variables

Current user

User variable

```
{% if user.is_authenticated %}
    <a href="{% url 'logout' %}">Logout
```



```
{% else %}
<a href="{% url 'login' %}">Login</a>
{% endif %}
```

Current User is logged & logout

```
</a>
```

Restricting Routes :-

localhost:8000/profile

Users → app

↳ views.py

```
def profilepage(request):
    return render(request, 'users/profile.html')
```

profile.html

```
<h1> {{ user.username }} </h1>
```

mainproject

```
path('profile/', user_views.profilepage,
     name='profile'),
```


base.html below Logout

```
<a href="{% url 'profile' %}">profile</a>
```

[Restriction of profile page]

add decorator

```
@login_required
def profilepage(request):
    return render(request, 'users/profile.html')
```

```
from django.contrib.auth.decorators
    import login_required
```

settings.py

```
LOGIN_URL = 'login'
```

Restricting Routes :-

localhost:8000/profile

Users → app

↳ views.py

```
def Profilepage(request):
    return render(request, 'users/profile.html')
```

Profile.html

```
<h1> {{ user.username }} </h1>
```

mainProject

```
path('profile/', user_views.profilepage,
     name='profile'),
```


base.html below Logout

```
<a href="{% url 'profile' %}">Profile</a>
```

[Restriction of profile page]

add decorator

```
@login_required
def profilepage(request):
    return render(request, 'users/profile.html')
```

```
from django.contrib.auth.decorators
    import login_required
```

settings.py

```
LOGIN_URL = 'login'
```

Restricting Routes :-

localhost:8000/profile

Users → app

↳ views.py

```
def Profilepage(request):
    return render(request, 'users/profile.html')
```

Profile.html

```
<h1> {{ user.username }} </h1>
```

mainProject

```
path('profile/', user_views.profilepage,
                 name='profile'),
```


base.html below Logout

```
<a href="{% url 'profile' %}">Profile</a>
```

Restriction of profile page

add decorator

```
@login_required
def profilepage(request):
    return render(request, 'users/profile.html')
```

```
from django.contrib.auth.decorators
    import login_required
```

settings.py

```
LOGIN_URL = 'login'
```

localhost:8000/profile/

→ login page

→ login

→ Profile page

Creating the Profile Model

Users

→ models.py

```
from django.contrib.auth.models import User

class Profile(models.Model):
    user = models.OneToOneField(User,
    image = models.ImageField(default='
    location = models.CharField(max_length=100)
```


linux/mac ← Sudo Pip install Pillow

Pip install Pillow / windows

→ open CMP as administrator

```
def __str__(self):
    return self.user.username
```

Python manage.py makemigrations

" " .py migrate

```
on_delete=models.CASCADE)
'profilepic.jpg', upload_to='Profile_
                              Pictures')
```

Python manage.py runserver

↳ login as admin

Register model

in admin.py

from .models import Profile

admin.site.register(Profile)

1) Adding Path To Upload Images
2) Adding the User Profile Picture
3) Setting up the Default Profile Picture

4) Add profile

Stop Server

Python manage.py shell

from django.contrib.auth.models import User

User = User.objects.filter(username='Sanjay').first()

User

↳ User.Profile

↳ User.Profile.image

↳ Profile_pictures/astrotrue.JPG

Main Project

Profile pictures

Setting.py

MEDIA_ROOT = os.path.join(BASE_DIR, 'pictures')

MEDIA_URL = '/pictures/'

exit() — Shell

Start server — Delete Previous

Add another Profile

Static file

(2) profile.html

<h1>{{user.username}}</h1>

<img src="{{user.profile.image.url}}">

Google

→ Uploading Static files in django

main Project

Urls.py

Paste

+=

width='100px' height='100px'

/ /

(3)

Register → newsiteUser

↳ demo@gmail.com

testtest456

↳ login

↳

/admin

Profiles →

newsiteUser → India

/login

↳ newsiteUser → testtest456

placeholder image

Google → IConfinder → User → Free

Download PNG

/ /

View Page Source → rename

↳ Pictures/profilepic.jpg

/ /

(3)

Register → newsiteuser

↳ demo@gmail.com

testtest456

↳ login

↳

/admin

Profiles →

newsiteUser → India

/login

↳ newsiteUser ↳ testtest456

placeholder image

Google → iConfinder → User → Free

Download PNG

/ /

View Page Source → filename

↳ Pictures/profilepic.jpg

Django Signals & Class Based Views :-

1. What are Django Signals
2. Implementing Django Signals
3. Class Based Views in Django
4. Implementing Class ~~Based~~ Detail View
5. Add User to Post
6. Adding Get ~~Absolute~~ URL Method
7. Automating User Association
8. Design Touchup: Login Form
9. Design Touchup: Register Page
10. Design Touchup: Add Item Page
11. Section Conclusion: what we learned
12. Source Code

1 U

Users

Signals.py

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import Profile
@receiver(post_save, sender=User)
def build_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()
```

apps.py

```
def ready(self):
    import users.signals
```

register page

Sampleuser Sample@gmail.com testpass

→ login → index page → Profile

Django Signals

User → Profile

Django Signals

Signal Dispatcher

Senders → Notification → Receivers

User Registeration → User Profile

Signal

apps.py

```
def ready(self):
    import users.signals
```

register page

Sampleuser Sample@gmail.com testtest

→ login → index page → Profile

Django Signals

User → Profile

Django Signals

Signal Dispatcher

Senders → Notification → Receivers

User Registeration → User Profile

Signal

1/1

③ Generic views

function based views (food, views.py)

Class based view.

Class based (list view :-

detail view

delete view

create view

1 Class based, list view

index view

```
from django.views.generic.list import
                    ListView

Class IndexClassView(ListView):
    model = Item;
    template_name = 'food/index.html'
    Context_object_name = 'item_list'


Path('', views.IndexClassView.as_view(),
        name='index'),
```

1 1

3 Generic views

function based views — food, views.py

Class based view.

Class based list view :-

detail view

delete view

Create view

1 Class based list view

index view

```
from django.views.generic.list import
                    ListView

Class IndexClassView(ListView):
    model = Item;
    template_name = 'food/index.html'
    Context_object_name = 'item_list'


Path('', views.IndexClassView.as_view(),
        name='index'),
```

detail-view

```
from django.views.generic.detail import
    DetailView

class FoodDetail(DetailView):
    model = Item
    template_name = 'food/detail.html'
```

```
path('<int:pk>/', views.FoodDetail.as_view(),
    name='detail'),
```

detail.html

Replace item with object

```
object.item_name
object.itemdesc
object.itemprice
```


Create View
Delete View
Update View

[5] which User has Posted :-

Model — Model — inbuilt

Item <~~> User (foreignKey)

in models.py (food)

```
from django.contrib.auth.models import
    User

User_name = models.ForeignKey(User,
    on_delete = models.CASCADE,
    default = 1)
```

```
Python manage.py makemigrations
Python manage.py sqlmigrate food 0003
Python manage.py migrate
```

1/1

detail-view

```
from django.views.generic.detail import
    DetailView

class FoodDetail(DetailView):
    model = Item
    template_name = 'food/detail.html'
```

```
path('<int:pk>/', views.FoodDetail.as_view(),
    name='detail'),
```

detail.html

Replace item with object

```
object.item_name
object.itemdesc
object.itemprice
```


1/1

Create View
Delete View
Update View

Q[5] Which User has Posted:-

Model ↔ Model - inbuilt

Item ↔ User (foreign key)

in models.py (food)

```
from django.contrib.auth.models import
    User

User_name = models.ForeignKey(User,
    on_delete = models.CASCADE,
    default = 1)
```

```
Python manage.py makemigrations
Python manage.py sqlmigrate food 0003
Python manage.py migrate
```

Python manage.py runserver

/admin (Manually)
↳ Add item

(6) from django.urls import reverse

(in models.py)

```
def get_absolute_url(self):
    return reverse("food:detail", kwargs
                   ={"pk": self.pk})
```

in Views.py

Convert it in class based view

```
# this is a class based views for Create
  item

from django.views.generic.edit import
  CreateView

Class CreateItem(CreateView):
    model = Item;
    fields = [`item_name', `item_desc',
              `item_price' `item_image']
    template_name = `food/item-form.html'
    def form_valid(self, form):
        form.instance.user_name = self.request.
                                  user

        return super().form_valid(form)
```

(Urls.py)

```
path('add', views.CreateItem.as_view(),
     name = `create_item'),
```

(logout) → Sampleuser (pass) → (add item)

www.itvedant.com

for checking login as admin

Sachin Superuser

again logout

register

brandnewuser new@gamil.com
test1test1947

login

add item

In Zorder.html

<h6>{{item.user_name}}</h6>

logout

login as admin

go in items

www.itvedant.com

Login form

{% extends 'food/base.html' %}

{% block body %}

(*) Paste

{% endblock %}

 offset-md-4

 bg-info text-white

login

www.itvedant.com

now paste

{{form}} instead

{% for field in form %}

{{field.name}} {{field}}

{% endfor %}

class = "btn-info"

Register Page Register.html

{% extends 'food/base.html' %}

{% block body %}
(*) Paste
{% endblock %}

Copy Same Code Login => Register

Delete ~~form~~ in Card-body
instead Paste registerform here

{{form}} {% for field in form %}

(*)

{% endfor %}

// /

```
<div class="form-group">
    {{field.name}}
    <br>
    {{field}}
</div>
```

"btn"
btn-info

Register / Signup

Add item page

Copy same code

```
<div class="form-group">
    {{field.name}}
    <br>
    {{field}}
</div>
```

"btn"

btn-info Register / Signup

~~Add item page~~

Copy same code