**HARSH KASLIWAL**

**2003085**

**C21**

**EXPERIMENT 11**

**AIM:** Django Web Framework

**Program 1:** Creating web application using Django web framework

* Installing Django
* Creating project
* Creating App and Views
* Creating and activating model
* Admin interface -Modify database from admin interface

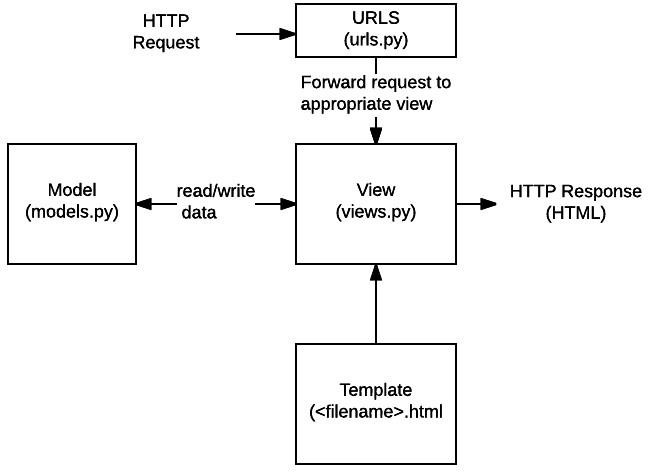
# Functions used:

1. [Djang](https://djangoproject.com/)o:

Django is a high-level Python Web framework that encourages rapid development and clean pragmatic design. A Web framework is a set of components that provide a standard way to develop websites fast and easily. Django’s primary goal is to ease the creation of complex database-driven websites. Some well known sites that use Django include PBS, Instagram, Disqus, Washington Times, Bitbucket and Mozilla.

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template.

Django web applications typically group the code that handles each of these steps into separate files:



* + URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data.
  + View: A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via models, and delegate the formatting of the response to templates.
  + Models: Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database.
  + Templates: A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A view can dynamically create an HTML page using an HTML template, populating it with data from a model. A template can be used to define the structure of any type of file; it doesn't have to be HTML!

**CODE -**

#manage.py import os import sys def main():

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'todo.settings') try:

from django.core.management import execute\_from\_command\_line except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you " "forget to activate a virtual environment?"

) from exc

execute\_from\_command\_line(sys.argv) if name == ' main ': main() #admin.py from django.contrib import admin from .models import \* admin.site.register(Task)

#apps.py from django.apps import AppConfig class TasksConfig(AppConfig):

name = 'tasks' #views.py from django.shortcuts import render, redirect from django.http import HttpResponse from

.models import \* from .forms import \* def index(request): tasks = Task.objects.all()

form = TaskForm() if request.method

=='POST':

form = TaskForm(request.POST) if form.is\_valid():

form.save() return redirect('/')

context = {'tasks':tasks, 'form':form} return render(request, 'tasks/list.html', context) def updateTask(request, pk): task = Task.objects.get(id=pk) form = TaskForm(instance=task) if request.method == 'POST': form = TaskForm(request.POST, instance=task) if form.is\_valid():

form.save() return redirect('/')

context = {'form':form} return render(request, 'tasks/update\_task.html', context) def deleteTask(request, pk): item = Task.objects.get(id=pk) if request.method == 'POST':

item.delete() return redirect('/') context =

{'item':item}

return render(request, 'tasks/delete.html', context) #models.py from django.db

import models class Task(models.Model):

title = models.CharField(max\_length=200) complete = models.BooleanField(default=False) created =

models.DateTimeField(auto\_now\_add=True) def

\_\_str (self): return self.title

#forms.py from django import forms from django.forms import ModelForm from .models import \* class TaskForm(forms.ModelForm):

title= forms.CharField(widget= forms.TextInput(attrs={'placeholder':'Add new task...'})) class Meta:

model = Task fields = ' all '

#urls.py from django.urls import path from . import views urlpatterns = [ path('', views.index, name="list"), path('update\_task/<str:pk>/',

views.updateTask, name="update\_task"), path('delete/<str:pk>/', views.deleteTask, name="delete"),

]

# OUTPUT –

