1 Create a new django project using command line

```
django-admin startproject projectname
```

2 Create a “Hello World” App in Django

```
In helloworld/views.py
from django.http import HttpResponse

def hello_world(request):
    return HttpResponse("Hello, World!")
```

```
In urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('helloworld/', include('helloworld.urls')),
]
```

3 Create a Django Form using forms.py

In forms.py

```
from django import forms
from django.shortcuts import render

class MyForm(forms.Form):
    name = forms.CharField(label='Name', max_length=100)
    email = forms.EmailField(label='Email')
    message = forms.CharField(label='Message', widget=forms.Textarea)


def my_view(request):
    if request.method == 'POST':
        form = MyForm(request.POST)
        if form.is_valid():
            # Form data is valid, process it here
            name = form.cleaned_data['name']
            email = form.cleaned_data['email']
            message = form.cleaned_data['message']
            # Additional processing or saving to the database
    else:
        form = MyForm()

    return render(request, 'my_template.html', {'form': form})

In templates/my_template.html
<form method="post" action="">
 {% csrf_token %}
 {{ form.as_p }}
 <button type="submit">Submit</button>
</form>
```

4 App to connect templates with models to serve data dynamically

```
In Models.py
from django.db import models

class MyModel(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

In Views.py
from django.shortcuts import render
from .models import MyModel

def my_view(request):
    data = MyModel.objects.all()
    return render(request, 'my_template.html', {'data': data})

In templates/my_template.html
{% for item in data %}
    <h2>{{ item.title }}</h2>
    <p>{{ item.description }}</p>
    <p>Created at: {{ item.created_at }}</p>
{% endfor %}
```

5 Write a Django web app to use parameters in Views.py

In views.py

```
from django.shortcuts import render
from django.http import HttpResponse

def greet(request, name):
   return HttpResponse(f"Hello, {name}!")


In url
http://localhost:8000/myapp/greet/noor/
```

6 Write a Django web app using control statements (If, for etc.)

```
In views.py
from django.shortcuts import render

def my_view(request):
   data = {
       'name': 'John Doe',
       'age': 25,
       'is_registered': True,
       'fruit_list': ['Apple', 'Banana', 'Orange', 'Grapes']
   }
   return render(request, 'my_template.html', {'data': data})

In templates/my_template.html
<!DOCTYPE html>
<html>
```

```
<head>
   <title>My App</title>
</head>
<body>
   <h1>Welcome, {{ data.name }}!</h1>

   {% if data.is_registered %}
       <p>You are a registered user.</p>
   {% else %}
       <p>Please register to access the full features.</p>
   {% endif %}

   <h2>Fruit List:</h2>
   <ul>
       {% for fruit in data.fruit_list %}
           <li>{{ fruit }}</li>
       {% endfor %}
   </ul>

   {% if data.age >= 18 %}
       <p>You are an adult.</p>
   {% else %}
       <p>You are a minor.</p>
   {% endif %}
</body>
</html>
```

7 Using blocks in Django Template and Extend base.html in Templates

`In myapp/templates/base.html`

```
<!DOCTYPE html>
<html>
<head>
   <title>{% block title %}{% endblock %}</title>
</head>
```

```
<body>
   <header>
       <h1>My App</h1>
   </header>

   <nav>
       <!-- Add your navigation links here -->
   </nav>

   <main>
       {% block content %}
       {% endblock %}
   </main>

   <footer>
       <p>&copy; 2023 My App. All rights reserved.</p>
   </footer>
</body>
</html>
```

In templates/home.html

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

{% block title %}Home{% endblock %}

{% block content %}
   <h2>Welcome to My App!</h2>
   <p>This is the home page.</p>
{% endblock %}
```

In Views.py

```
from django.shortcuts import render

def home(request):
   return render(request, 'home.html')
```

8 Work with Django Template built in Tags and Filter

In templates/my_template.html

```
<!DOCTYPE html>
<html>
<head>
   <title>My Template</title>
</head>
<body>
   {% if items %}
       <ul>
           {% for item in items %}
               <li>{{ item|capfirst }}</li>
           {% endfor %}
       </ul>
   {% else %}
       <p>No items available.</p>
   {% endif %}

   <p>Total items: {{ items|length }}</p>

   <p>Today's date: {{ today|date:"F j, Y" }}</p>

   <p>Joined names: {{ names|join:", " }}</p>
</body>
</html>
```

In views.py

```
from django.shortcuts import render

def my_view(request):
   data = {
       'items': ['apple', 'banana', 'orange'],
       'today': datetime.date.today(),
       'names': ['John', 'Jane', 'Tom']
   }
   return render(request, 'my_template.html', {'data': data})
```

9 Handling 404, 502 pages in Django

In views.py

```python
from django.shortcuts import render

def error_404_view(request, exception):
   return render(request, '404.html', status=404)

def error_502_view(request):
   return render(request, '502.html', status=502)
```

In 404.html

```html
<!DOCTYPE html>
<html>
<head>
   <title>404 Page Not Found</title>
</head>
<body>
   <h1>Page Not Found</h1>
   <p>The requested page could not be found.</p>
</body>
</html>
```

In 502.html

```html
<!DOCTYPE html>
<html>
<head>
   <title>502 Bad Gateway</title>
</head>
<body>
   <h1>Bad Gateway</h1>
   <p>The server encountered a temporary error and could not complete the request.</p>
</body>
</html>
```

In urls.py

```python
from django.urls import path
from . import views
```

```
handler404 = 'myapp.views.error_404_view'
handler502 = 'myapp.views.error_502_view'

urlpatterns = [
   # Add your app URLs here
]
```

10 Create a Django model called “Book” with fields for title, author, publication date, and ISBN. Write the necessary code to migrate the model to the database and ensure it is correctly reflected in the database schema.

```
from django.db import models

class Book(models.Model):
   title = models.CharField(max_length=200)
   author = models.CharField(max_length=100)
   publication_date = models.DateField()
   isbn = models.CharField(max_length=13)

   def __str__(self):
       return self.title
```

```
python manage.py makemigrations
python manage.py migrate
```

11 Implement a Django view that displays a list of all books in the database. The view should render a template that shows the title and author of each book. Write the necessary code to define the view, map it to a URL, and create the corresponding template to display the book list.

```
In views.py
from django.shortcuts import render
from .models import Book

def book_list(request):
   books = Book.objects.all()
   return render(request, 'book_list.html', {'books': books})
```

```
In templates/book_list.html
<!DOCTYPE html>
<html>
<head>
   <title>Book List</title>
</head>
<body>
   <h1>Book List</h1>
   <ul>
       {% for book in books %}
           <li>{{ book.title }} - {{ book.author }}</li>
       {% endfor %}
   </ul>
</body>
</html>
```

12 Create a Django form that allows users to add new books to the database. The form should include fields for the title, author, publication date, and ISBN. Implement a view that handles form submissions, validates the data, and saves the new book to the database.

```
In forms.py
from django import forms
from .models import Book

class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        fields = ['title', 'author', 'publication_date', 'isbn']


In views.py
from django.shortcuts import render, redirect
from .forms import BookForm

def add_book(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm()
    return render(request, 'add_book.html', {'form': form})


In templates/add_book.html
<!DOCTYPE html>
<html>
<head>
    <title>Add Book</title>
</head>
<body>
    <h1>Add Book</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Add Book</button>
    </form>
</body>
</html>
```

```
In urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('books/add/', views.add_book, name='add_book'),
]
```

13 Implement a search functionality in Django that allows users to search for books by title or author. Create a search form and a view that retrieves matching books from the database and displays them in a template.

```
In forms.py
from django import forms

class SearchForm(forms.Form):
    search_term = forms.CharField(label='Search')

In views.py
from django.shortcuts import render
from .forms import SearchForm
from .models import Book

def search_books(request):
    form = SearchForm(request.GET)
    if form.is_valid():
        search_term = form.cleaned_data['search_term']
        books = Book.objects.filter(title__icontains=search_term) |
Book.objects.filter(author__icontains=search_term)
    else:
        books = Book.objects.none()
    return render(request, 'search_books.html', {'form': form, 'books':
books})
```

In templates/search_books.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Search Books</title>
</head>
<body>
    <h1>Search Books</h1>
    <form method="get">
        {{ form }}
        <button type="submit">Search</button>
    </form>
    <hr>
    <h2>Search Results</h2>
    {% if books %}
        <ul>
            {% for book in books %}
                <li>{{ book.title }} - {{ book.author }}</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>No matching books found.</p>
    {% endif %}
</body>
</html>
```

14 Create a Django model called Category that represents different book categories. Establish a many-to-many relationship between the Book and Category models, allowing books to belong to multiple categories.

```
In models.py
from django.db import models

class Category(models.Model):
   name = models.CharField(max_length=100)

   def __str__(self):
       return self.name

class Book(models.Model):
   title = models.CharField(max_length=200)
   author = models.CharField(max_length=100)
   publication_date = models.DateField()
   isbn = models.CharField(max_length=13)
   categories = models.ManyToManyField(Category)

   def __str__(self):
       return self.title


python manage.py makemigrations
python manage.py migrate
```

15 Implement user authentication in Django by creating a registration form, login form, and logout functionality. Write the necessary views, templates, and URL mappings to allow users to register, login, and logout.

```
In views.py
```

```python
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm
from django.contrib.auth import login, logout
from django.shortcuts import render, redirect

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = UserCreationForm()
    return render(request, 'register.html', {'form': form})

def user_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('home')
    else:
        form = AuthenticationForm()
    return render(request, 'login.html', {'form': form})

def user_logout(request):
    logout(request)
    return redirect('home')
```

In templates/register.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Registration</title>
</head>
<body>
    <h1>Registration</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Register</button>
    </form>
```

```
</body>
</html>
```

In login.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Login</title>
</head>
<body>
   <h1>Login</h1>
   <form method="post">
       {% csrf_token %}
       {{ form.as_p }}
       <button type="submit">Login</button>
   </form>
</body>
</html>
```

In urls.py

```
from django.urls import path
from . import views

urlpatterns = [
   path('register/', views.register, name='register'),
   path('login/', views.user_login, name='login'),
   path('logout/', views.user_logout, name='logout'),
]
```

16 Create a view that requires authentication, such as a user profile page. Ensure that only authenticated users can access the protected view and redirect unauthenticated users to the login page.

```
In views.py
from django.contrib.auth.decorators import login_required
from django.shortcuts import render

@login_required
def profile(request):
   return render(request, 'profile.html')

In templates/profile.html
<!DOCTYPE html>
<html>
<head>
   <title>User Profile</title>
</head>
<body>
   <h1>User Profile</h1>
   <p>Welcome, {{ request.user.username }}!</p>
   <!-- Display user-specific profile information -->
</body>
</html>
```

17 Implement a rating system for books using Django’s built-in authentication system. Allow users to rate books on a scale of 1 to 5 and display the average rating for each book.

```
In models.py
from django.db import models
```

```
class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=100)
    publication_date = models.DateField()
    isbn = models.CharField(max_length=13)
    average_rating = models.DecimalField(max_digits=3, decimal_places=2,
default=0)

    def __str__(self):
        return self.title

from django.contrib.auth.models import User

class Rating(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE,
related_name='ratings')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    rating = models.PositiveIntegerField(choices=[(1, '1'), (2, '2'),
(3, '3'), (4, '4'), (5, '5')])

    def __str__(self):
        return f"{self.user.username}'s rating for {self.book.title}"
```

In views.py

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import get_object_or_404, redirect, render
from .models import Book, Rating

@login_required
def rate_book(request, book_id):
    book = get_object_or_404(Book, id=book_id)
    if request.method == 'POST':
        rating_value = int(request.POST.get('rating'))
        if rating_value in range(1, 6):
            Rating.objects.update_or_create(
                book=book,
                user=request.user,
                defaults={'rating': rating_value}
            )
    return redirect('book_detail', book_id=book_id)
```

```
python manage.py makemigrations
python manage.py migrate
```

18 Create a custom Django template filter or tag that performs a specific operation, such as formatting a date or applying a custom text transformation. Use the custom filter or tag in one of the templates and verify that it produces the expected output.

```
In custom_filters.py
from django import template

register = template.Library()

@register.filter
def capitalize_first(value):
   return value.capitalize()

In templates/my_template.html

{% load custom_filters %}

<!DOCTYPE html>
<html>
<head>
   <title>Book Details</title>
</head>
<body>
   <h1>Book Details</h1>
   <p>Title: {{ book.title|capitalize_first }}</p>
</body>
</html>

In views.py
from django.shortcuts import render

def book_details(request):
   book = {
       'title': 'the great gatsby'
   }
   return render(request, 'my_template.html', {'book': book})
```

19 Implement file uploads and storage in Django. Create a model that includes a FileField or ImageField, allowing users to upload files or images. Configure a file storage backend, such as local storage ensure that uploaded files are saved correctly and accessible.

Update in settings.py
# Specify the desired location for uploaded files
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Specify the URL prefix for accessing uploaded files
MEDIA_URL = '/media/'

```
In models.py
from django.db import models

class MyModel(models.Model):
    file = models.FileField(upload_to='files/')

In forms.py
from django import forms
from .models import MyModel

class MyModelForm(forms.ModelForm):
    class Meta:
        model = MyModel
        fields = ['file']
```

In views.py

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

def upload_file(request):
   if request.method == 'POST':
       form = MyModelForm(request.POST, request.FILES)
       if form.is_valid():
           form.save()
           return redirect('upload_success')
   else:
       form = MyModelForm()
   return render(request, 'upload.html', {'form': form})

def upload_success(request):
   return render(request, 'upload_success.html')
```

In templates/upload.html

```
<form method="post" enctype="multipart/form-data">
 {% csrf_token %}
 {{ form.as_p }}
 <button type="submit">Upload</button>
</form>
```

In templates/upload_success.html

```
<h1>File uploaded successfully!</h1>
```

In urls.py

```
from django.urls import path
from . import views

urlpatterns = [
   path('upload/', views.upload_file, name='upload_file'),
   path('upload/success/', views.upload_success,
name='upload_success'),
]
```

20 Implement form validation using Django’s built-in form validation and validation constraints. Add custom validation logic to the form fields to ensure that certain conditions are met when users submit the form.

```
In models.py
from django import forms
from .models import MyModel

class MyModelForm(forms.ModelForm):
    class Meta:
        model = MyModel
        fields = ['name', 'email', 'password']

    def clean_name(self):
        name = self.cleaned_data['name']
        # Add custom validation logic for the name field
        if len(name) < 3:
            raise forms.ValidationError("Name should have at least 3
characters.")
        return name

In views.py
from django.shortcuts import render, redirect
from .forms import MyModelForm

def my_form_view(request):
    if request.method == 'POST':
        form = MyModelForm(request.POST)
        if form.is_valid():
            # Perform necessary actions with the validated form data
```

```
           name = form.cleaned_data['name']
           email = form.cleaned_data['email']
           password = form.cleaned_data['password']
           # ...
           return redirect('success')
   else:
       form = MyModelForm()
   return render(request, 'my_form.html', {'form': form})
```

In templates/my_form.html

```
<form method="post">
 {% csrf_token %}
 {{ form.as_p }}
 <button type="submit">Submit</button>
</form>
```

In templates/success.html

```
<h1>Form submitted successfully!</h1>
```

In urls.py

```
from django.urls import path
from . import views

urlpatterns = [
   path('form/', views.my_form_view, name='my_form_view'),
   path('success/', views.success_view, name='success'),
]
```