27/5/24

Assignment - 1

Sudhanshu
1JS21CS155

01

1) Define full stack development. Explain the different popular stacks for the development of web application.

Ans) Full stack development refers to the development of both front end (client side) & back end (server side) portions of web application.

Popular Stacks

* MEAN stack : MongoDB, Express, AngularJS and NodeJS.
* MERN Stack : MongoDB, Express, ReactJS and NodeJS.
* Django Stack : Django, Python & MySQL Database.
* Rails (or) Ruby on Rails : Uses HTML, Ruby, Rails & MySQL.
* LAMP stack : Linux, Apache, MySQL, PHP.

---

2) List & explain the key advantage of full stack development

Ans) Key advantages of Full Stack Development are:

1) Saves time and money.
2) Rounded Solution
3) Great Exposure
4) Complete Ownership
5) Greater opportunity with more learning.

02 3) Define the following:

a) Django:

↳ It is a free and open source web application framework which offers fast & effective dynamic website development.

b) Framework:

↳ A web framework is a code library that makes web development faster and easier by providing common patterns for building reliable, scalable and maintainable web applications.

c) Template:

↳ A template is a text file that defines the structure or layout of a document with placeholders for the dynamic content. It allows developers to separate the presentation layer from the business logic by embedding DTL within the HTML. This enables the dynamic rendering of content based on the context provided by Django views.

4) Write a simple Common Gateway Interface (CGI) in Python that displays the 10 most recently published books from a database.

Ans). #! /usr/bin/python

```
import MySQLdb

print "Content-Type: text/html"
print
print "<html><head><title>Books</title></head>"
print "<body>"
print "<h1>Books:</h1>"
print "<ul>"

connection = MySQLdb.connect(user='me', passwd='letmein',
        db='my-db')
cursor = connection.cursor()
cursor.execute('SELECT name FROM books ORDER BY
            pub-date DESC LIMIT 10")
for row in cursor.fetchall():
    print "<li> %s </li>" % row[0]
print "</ul>"
print "</body></html>"
connection.close()
```

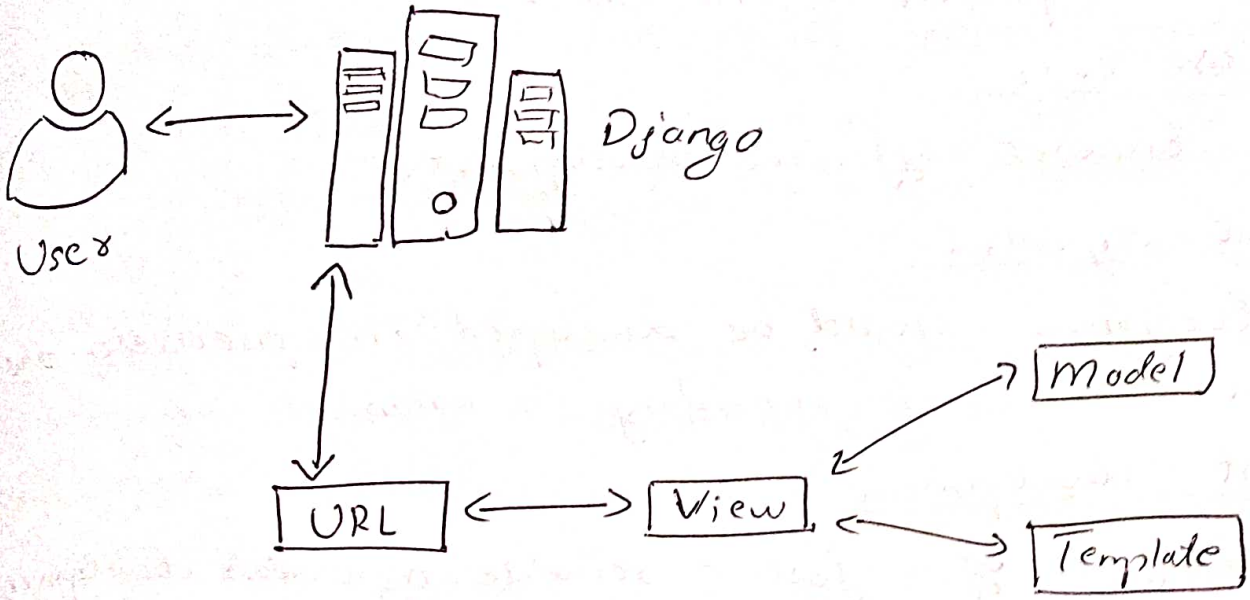
04 5) List & explain the limitations of CGI standards.

Ans) Limitations of CGI standards are:

1) Performance overhead
   a) Process creation
   b) Resource Intensive
2) Scalability issues.
   a) ~~Input~~ Limited Concurrency
   b) High Memory usage
3) Security Concerns
   a) Input validation
   b) File Permissions
4) State Management
5) Maintenance & Debugging
   a) Complexity
   b) Error Handling
6) Probability Issues
   a) Platform Dependency
   b) Environment Configuration
7) Lack of Modern Features
   a) Asynchronous Processing
   b) Limited Language Support.

6) With a neat diagram explain the architecture of Django MVT. 05

Ans) Flow of Control in MVT



- User sends a requests for a resource to the Django.
- Django works as a controller & check the available resource in URL.
- If URL is mapped view is called.
- It interacts with model & template.
- Then it renders a template.
- At last Django responds back to the user & sends it a template as a response.

06 7) List & explain characteristics of Django.

Ans) a) Loosely Coupled

↳ Django helps you to make each element of its stack independent of the others.

b) Less code

↳ Ensures effective development.

c) Not repeated

↳ Everything should be developed in precisely one place instead of repeating it again.

d) Fast development

↳ Django's offers fast & reliable application development.

e) Consistent Design

↳ Django maintains a clean design & makes it easy to follow the best web development practices.

---

8) Write a Django script, that displays the ten most recently published books from a database. Explain the functional features of view.py, models.py & urls.py.

Ans) # models.py (database tables)

```
from django.db import models

class Book(models.Model):
```

```
name = models.CharField(max_length = 50)
pub_date = models.DateField()
```

- This models.py file contains a description of database table as a python class., it is called as Model.
- Using this class, we can create, retrieve, update & delete records in database using simple python code rather than writing repetative SQL statements.

views.py (business logic)

```
from django.shortcuts import render_to_response
from models import Book
def latest_Books(request):
    book_list = Book.objects.order_by('pub_date')[:10]
    return render_to_response('latest_books.html', {'book_list': book_list})
```

- This views.py file contains the business logic for the page, in the latest_books() function.

urls.py (URL config)

```
from django.conf.urls.defaults import *
import views
urlpatterns = patterns('',
    (r'latest/$', views.latest_books),
)
```

- This file specifies which view is called for a given URL pattern.
- In this case URL /latest/ will be handled by function.

08 # latest_books.html (template)

```
<html>
<head>
  <title> Books </title>
</head>
<body>
    <h1> Books </h1>
    <ul>
        {% for book in book_list %}
        <li> {{ book.name }} </li>
        {% endfor %}
    </ul>
</body>
</html>
```

---

4) With an example program explain concept of mapping URLs to view.

Ans) • A URL config is like a table of contents for your Django-powered web site.

- It's a mapping b/w URL patterns & view functions that should be called for those URL patterns.
- Django URL are cleaner simple so called Pretty URLs.
- URLs are created using regular expressions.
- When user requests a URL, django search in the list of URL patterns defined in urls.py & finds the first match, if not match returns not found Error.
- The urls.py provides a way to map a URL to a view.

- To map a URL to a view we can create urlpatterns list in urls.py whose each element of the list is a path() or repath() call that maps a URL pattern to a view.

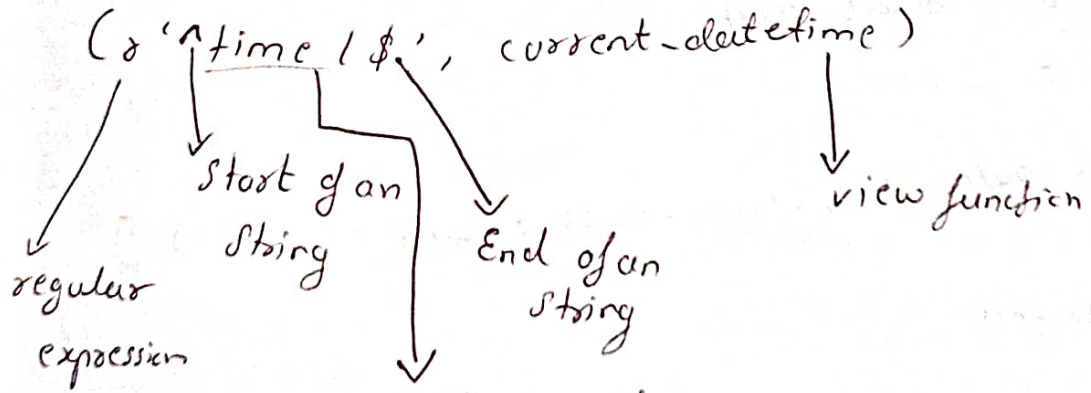
example:

```
1 | from django.conf.urls.defaults import *
2 | from app.views import view function
3 | urlpatterns = patterns ('', (r'^time/$', current_datetime))
```

- First line : imports all objects from django.conf.urls ~~defaults~~ including patterns.
- Second line : imports view functions from apps.
- Third line : Function called patterns() & saves the result into a variable called urlpatterns.

-> This variable defines the mapping blw URL & the code that handles these URLs. Also expect to find ROOT_URLCONF.

->



10 11) Develop a python program to create an Django app that displays current date & time 4 hours ahead & 4 hours before as an offset of current date & time in server.

Ans) #views.py

```
from django.shortcuts import render, HttpResponse
from datetime import datetime, timedelta

def date(request):
    current = datetime.now()
    before  = current - timedelta(hours=4)
    after   = current + timedelta(hours=4)
    return HttpResponse(f'<center><h1>Date & time in server
    4 hours ahead : <br> {after} <br> Date & time in server
    4 hours behind : <br> {before} <br></h1>')
```

#urls.py -> app

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

urlpatterns = [ path('', views.date, name="date"), ]
```

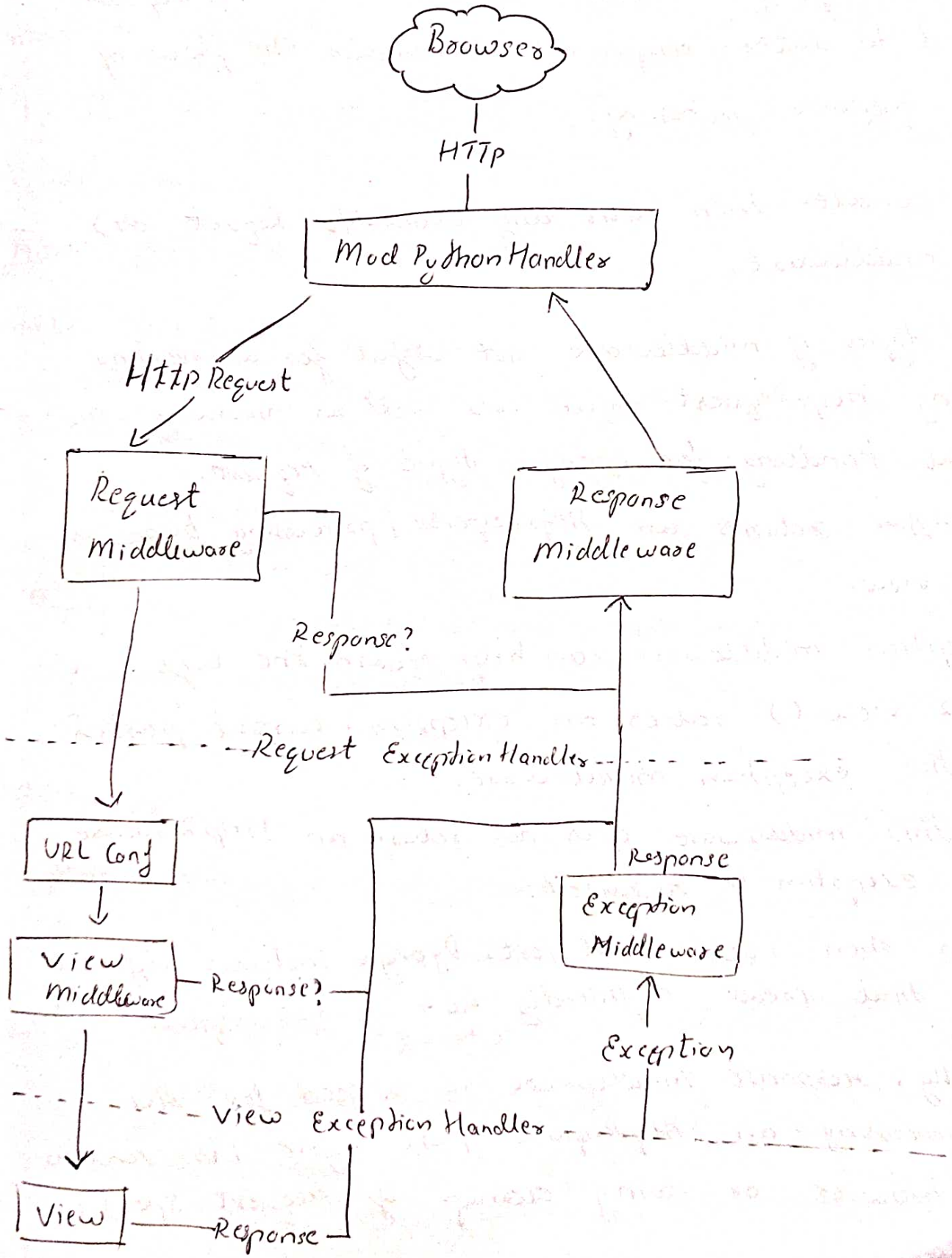
#urls.py -> project

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

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

12) With a neat flow diagram explain the request response mechanisms of Django.

Ans)



12 • When an HTTP request comes in from the browser, a server-specific handler constructs the HttpRequest passed to later components & handles the flow of the response processing.

- The handler then calls any available Request (or) view middleware.
- These types of middleware are useful for augmenting incoming HttpRequest objects as well as providing the special handling for specific types of requests.
- If either returns an HttpResponse, processing bypasses the view.
- Exception middleware can help squash the bugs.
- If a view() raises an exception, control passes to the exception middleware.
- If this middleware does not return an HttpResponse, the exception is reraised.
- Even then, all is not lost. Django includes default views that create a friendly 404 & 500 response.
- Finally, response middleware is a good for the postprocessing an HttpResponse just before it's sent to the browser or doing cleanup of request specific resources.

13) Apply the wildcard URL patterns & write a Django script that displays the 10 most recently published books from a database.

Ans) 1) Set up Django Project & App

2) # models.py

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    pub_date = models.DateField()

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

3) Apply the migrations to create table in DB

```
python manage.py makemigrations books ---> app
python manage.py migrate
```

4) # views.py

```
from django.shortcuts import render
from .models import Book

def recent_books(request):
    books = Book.objects.order_by('-pub_date')[:10]
    return render(request, 'books/recent_books.html',
                  {'books': books})
```

5) # urls.py → app

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

urlpatterns = [
    path('recent/', views.recent-books, name='
]                                    recent-books'),
```

6) # urls.py → project

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

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

7) Temple creation for books // recent-books.html

```
<!DOCTYPE html>
<head>
    <title> Recent Books </title>
</head>
<body>
    <h1> 10 most Recently Published books </h1>
    <ul>
        {% for book in books %}
            <li> {{ book.title }} - Published on {{ book.
        {% endfor %} pub-date }} </li>
    </ul>
</body>
</html>
```

8) Python manage.py runserver.

14) Create a template code that describes an HTML page that thanks a person for placing an order within a company.

Ans)

```
<html>
<head>
  <title> Ordering notice </title>
</head>
<body>
  <p> Dear {{ person_name }}, </p>
  <p> Thanks for placing an order from {{ company }}. It is
    scheduled to ship on {{ ship_date | date: "F j, Y" }}. </p>
  <p> Here are the items you've ordered: </p>
  <ul>
    {% for item in item_list %}
    <li> {{ item }} </li>
    {% endfor %}
  </ul>
  {% if ordered_warranty %}
    <p> Your warranty information will be included in the
        package. </p>
  {% endif %}
  <p> Sincerely, <br /> {{ company }} </p>
</body>
</html>
```

16 15) List & explain the steps in using the template system.

Ans)

1) Create a Django Project and App.
2) Configure Template Directory.
   ↳ modify the setting.py to include the templates directory in the 'TEMPLATES' setting.
3) Create Template Files
4) Create views to Render templates.
5) Configure URLs
6) Include App URLs in Project's 'urls.py'.

---

16) With on exple progrom explain the concept of multiple contexts, same template.

Ans)
- In a single template object we can render() multiple contexts ()
- for name in ('John', 'Julie', 'DNR'):
  t = Template ("Hello, {{ name }}")
  print t.render (context ({'name': name}))
- Here the template is called multiple times for different names.
- Instead multiple contexts in same templates as shown below.

```
t = Template ("Hello, {{name}}") → Single call to template
for name in ('BNR', 'RNR', 'SNR'):
    print t.render (Context ({'name': name}))
```

↳ calling render multiple times on it.

---

17) With an example program explain the how invalid variables are handled.

Sol) • By default if variable doesn't exist, template system renders a empty string.

ex:- from django.template import Template, Context

```
t = Template ('Your name is {{name}}.')
t.render (context())
```

o/p => 'Your name is.'

→ t.render (Context ({'var': 'hello'}))

↳ not defined variables.

o/p => 'Your name is.'

→ t.render (Context ({'NAME': 'hello'}))

o/p => 'Your name is.'

• Because of above issue lookups failed & webpage becomes inaccessible due to small syntax error.

- In Django the above problem can be overcome by setting the Django config by implement a custom template filter or template tag.
- This approach allows us to handle cases where variables are not found in the template context and provide a fallback behaviour instead of returning an empty string.

---

(8) With an example program explain the basic template tags and filters.

Ans) Tags

1) if/else

syntax: {% if %} } Displays everything
{% endif %} } b/w if it is true.

example:

```
{% if today_is_weekend %}
    <p> Enjoy </p>
{% else %}
    <p> come to work </p>
{% endif %}
```

- empty list [], tuple (), dictionary {}, string '', zero, none, are all false in boolean context.

- if tag accepts and, or, not.

ex:- {% if stud_list and teacher_list %}
{% endif %}

- if tag doesn't allow and & or clauses within the same tag because order of logic would be ambigious.
- we can combine and & or by advanced logic by using nested {% if %} tags.

2) for

syntax : {% for %}
{% endfor %}

- The tag allows you to loop over each time in a sequence.
- ex:-
{% for item in item_list %}
 {{ item.name }}
{% endfor %}

- To reverse the order :
{% for item in item_list reverse %}
{% endfor %}
- We can nest for..
- {% for %} set forloop variable which has attributes which gives the progress of the loop.

3) ifequal / ifnotequal.

- The {% ifequal %} tag compares 2 values and displays everything blw {% ifequal %} and {% endifequal %} if the values are equal.
- {% ifnotequal %} is when values are not equal.
- The arguments can be with single or double quotes.
- Only template variables, strings, integers & decimal numbers are allowed as arguments.
- Python dictionaries, lists and booleans can't be used.

ex:- {% ifequal section 'sitenews' %}
<h1> Site News </h1>
{% else %}
<h1> No News Here </h1>
{% endifequal %}

4) Comments

=) {# This is a comment #}

- The comment will not be output when the template is rendered.
- A comment cannot span multiple lines.

Filters

- Filters are simple ways of altering the values of variables before they are displayed.
- ex:- {{name | lower}}
  - ↳ converts text to lowercase.
- Use a pipe (|) to apply a filter.
- Filters can be chained.

  ex:- {{ my_text | escape | linebreaks }}
- Some filters take arguments.

  ex:- {{bio | truncatewords : "30" }}