DJANGO WEEK 3

1. What is IDE and its use in Web Development?

—> IDE (Integrated Development Environment) can be defined as software that gives its users an environment for performing programming, along with development as well as testing and debugging the application.

—> Examples of IDE are :

- Visual Studio Code
- Eclipse
- Oracle NetBeans
- Xcode, etc

2. What is Django?

—> Django is a Python web framework for developing applications.

—> It is secure, Versatile, portable and easy to maintain.

—> It is a high-level Python web framework that allows the rapid development of secure and maintainable websites.

—> It is free and open-source and used for full-stack and server development.

—> It is suitable for the backend as well as the front-end.

—> It is a collection of Python libraries that allows us to develop useful web apps ideal for backend and frontend purposes.

—> It follows the MVT (Model, View and Template) Architectural pattern.

—> **Model :** It is defined as the structure of stored data, including the field types, their maximum size, default values, etc. Each model maps to a single database table.

—> **Template :** It deals with the presentation of data.

—> **View :** It is a component that includes the logic that will be displayed to the user, usually represented by HTML, CSS or JavaScript files.

—> To install django we must have Python installed and version 3.4

—> Install it using python -m pip install Django

Syntax :

python -m django startproject <Project_Name>

python manage.py runserver

python manage.py startapp <App_Name>

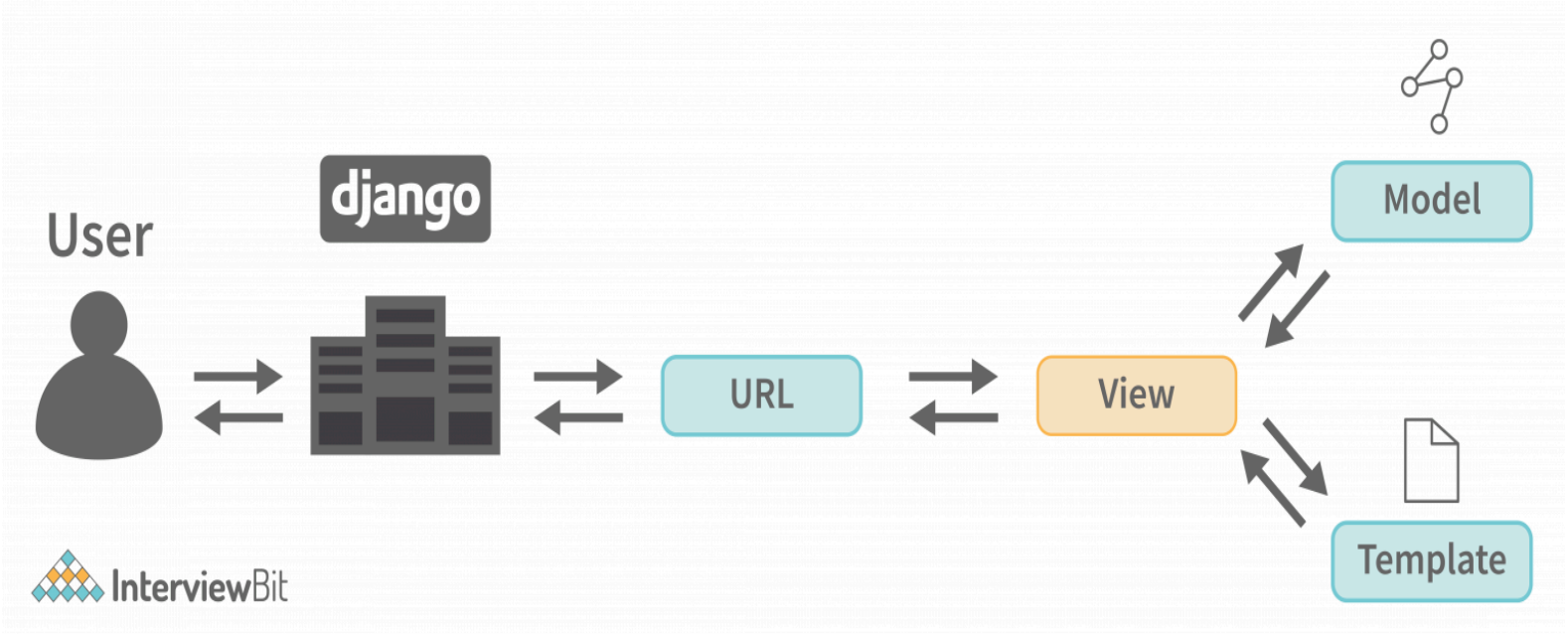
python manage.py runserver

DJANGO WEEK 3

3. What is Django Architecture?

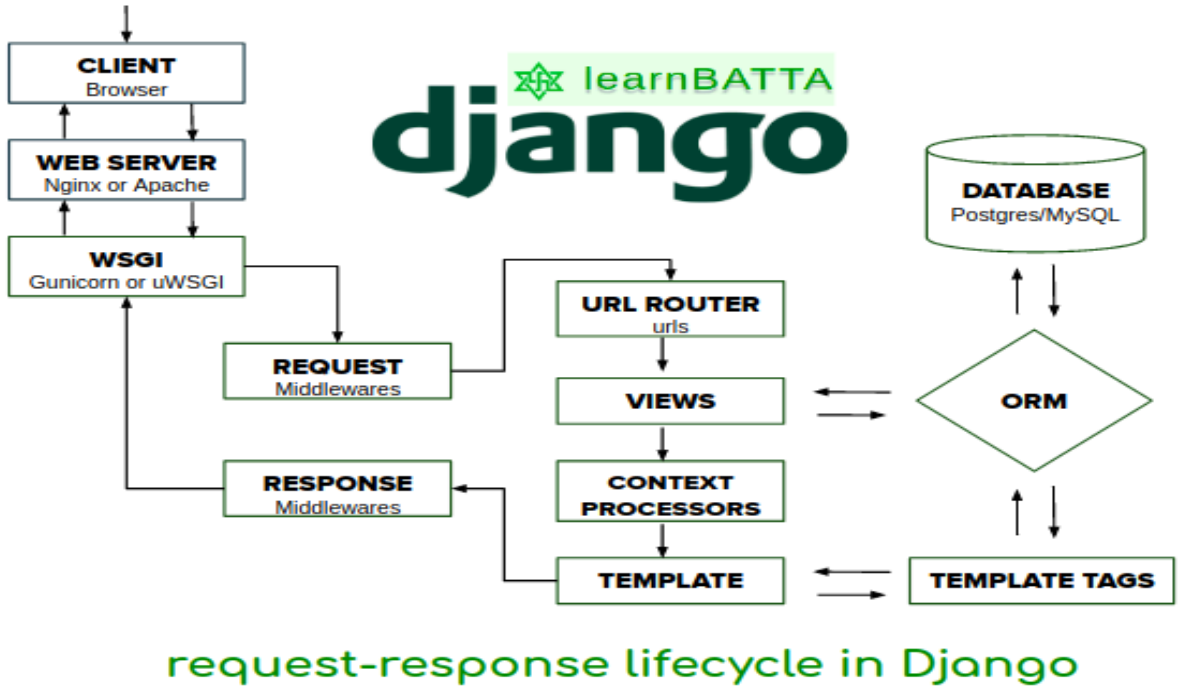
```
+---firstapp
|   |   admin.py
|   |   apps.py
|   |   forms.py
|   |   models.py
|   |   tests.py
|   |   urls.py
|   |   views copy.py
|   |   views.py
|   |   __init__.py
|   |
|   +---migrations
|   |       0001_initial.py
|   |       __init__.py
|   |
|   \---templates
|       \---firstapp
|               home.html
|
```

SCALER Topics



DJANGO WEEK 3

4. How does request/response work and what is it life cycle?



—> When a client sends a request to the server it first passed to the webserver.
—> Webserver contains the configuration rules to dispatch the request to the WSGI server or served by itself.
—> WSGI server pass the request to the django application.

LAYERS OF DJANGO APPLICATION :

1. Request Middlewares
2. URL Router or URL Dispatcher
3. Views
4. Context Processors
5. Template Renderers
6. Response Middlewares

1. Request Middlewares :

—> Whenever the request comes in, it is handled by the request middlewares.
—> We can have multiple middlewares. We can find it in project settings(settings.py).
—> Django request middlewares follows the order while processing the request.
—> Suppose, if we have request middlewares in the order A, B, C, then the request first processed by the middleware A and then B and then C.
—> Django comes up with bunch of default middlewares. We can also write our own or custom middlewares.

DJANGO WEEK 3

2. URL Router :
—> After request processed by the middlewares it will be submitted to the URL Router.
—> It will take the request from the request middlewares and it takes the URL Path from the request.
—> Based on the url path URL router will tries to match the request path with the available URL Patterns.
—> These URL Patterns are in the form of regular expressions.
—> After matching the URL path with available URL patterns the request will be submitted to the view which is associated with the URL.

3. Views :
—> Views processes the logic using request and request data (data sent in POST, GET, etc).
—> After processing the request in the view the request is sent context processors

4. Context Processors :
—> Request context processors adds the additional context data as defined.
—> the context used by renderers to render the template to generate the HTTP response.

5. Template Renderers :
—> Template renderers use the context generates the response.
—> The response content can be XML, HTML, JSON, etc.

6. Response Middlewares :
—> Again the request will send back to the Response middlewares to process it.
—> Response middlewares will process the request and adds or modifies the header/body information before sending it to the client.
—> client or Browser receives the response and shows it to the user.

5. What is a Virtual Environment?
—> Virtual Environments are a way to create isolated Python environments with their own Python interpreter and package directories.
—> They allow you to install packages and dependencies without affecting other projects on the same machine.
—> To create a virtual environment, you can use the ‘venv’ module that comes with Python 3.
pip install virtualenv
virtual venv

—> Once created, you can activate the virtual environment using a script for your shell
source venv/bin/activate

—> To deactivate the virtual environment, you can use the ‘deactivate’ command.
deactivate

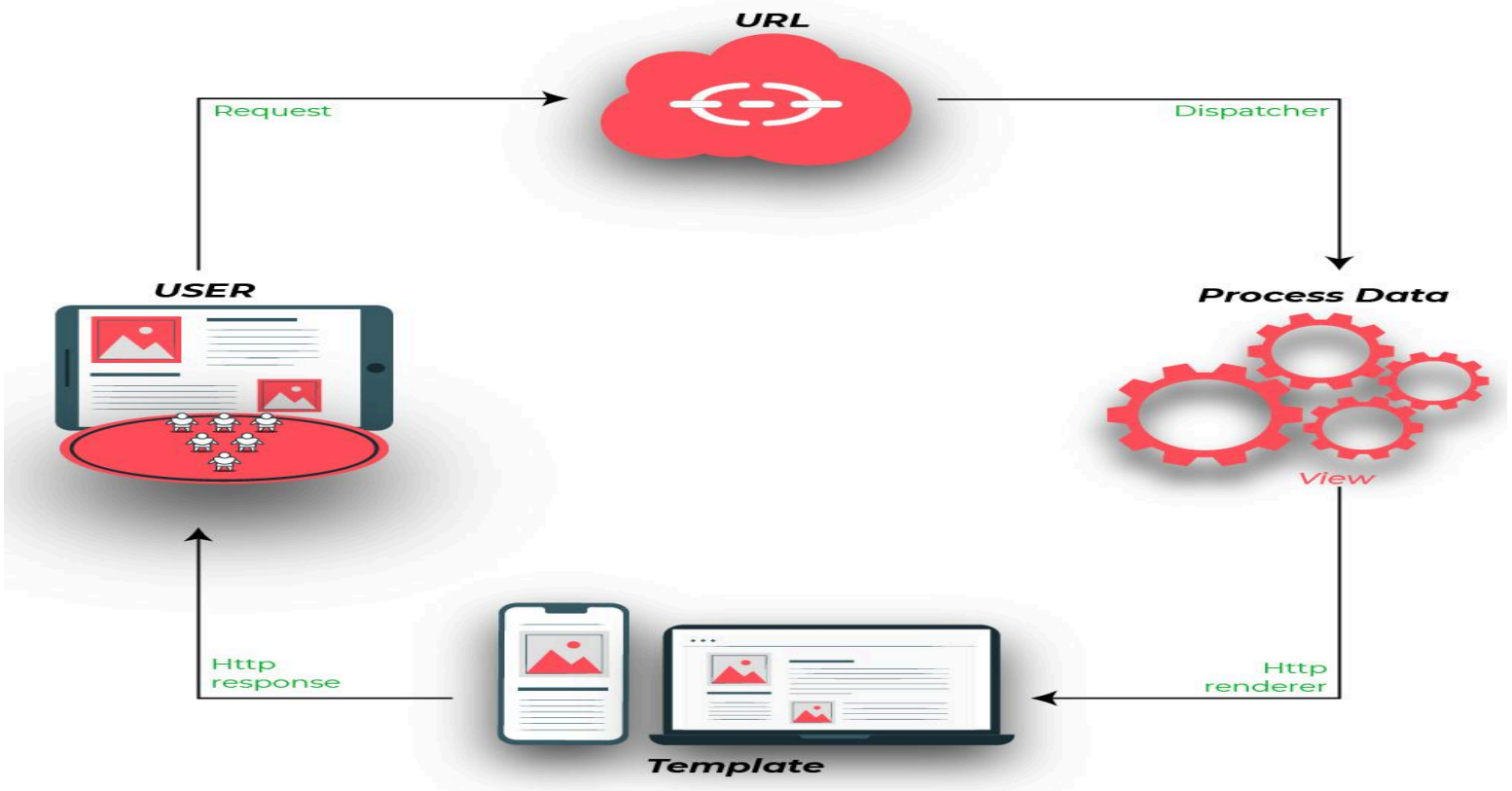
DJANGO WEEK 3

6. What is MVT in Django?
—> The MVT stands for (Model, View and Templates)

Model :
—> Django applications use Python objects called models to access and manage data. These models define the internal structure of the storing data, like field types and their maximum size, default values, label text for forms, etc. Models creation is independent of which database you are using. You can use any Database. You don't need to talk to the database directly. You only need to create your model structure and other code.

View :
—> A view function in Django is a python function that accepts a Web requests and delivers a Wen response. The response in Django views can be anything that a web browser can show, including the HTML of a Web page, a redirect, an XML document, a 404 error, an image, etc. Views in Django are part of the user interface in Django application because they return the web pages which contain HTML/CSS/JAVASCRIPT in the template of the Django.

Template :
—> Template in Django contains the static content of a django project like HTML, CSS and JAVASCRIPT along with the image used in the project. We can set the path of the Django template from the setting.py file of the Django project.



DJANGO WEEK 3

7. How to create Django Project?
—> Follow this Syntax to create Django Project :
python -m django startproject <Project_Name>
—> If we want to check objects of Django write :
python -m django
python manage.py runserver

8. How to create app in Django Project?
—> Follow this Syntax to create Django App in Django Project :
python -m django startproject <Project_Name>
cd <Project_Name>
—> Now, we can see that in our project there are some files like : __init__.py, asgi.py, settings.py, urls.py, wsgi.py, manage.py
i. __init__.py : It can contain initialization code for the package or it can be an empty file.

ii. wsgi.py : Web Server Gateway Interface is used for communication between web servers and Python Web Applications or frameworks.
—> Once your application is live, there can be thousands of requests in your application and it is capable of serving thousands of requests at a time.
—> Several servers support WSGI : GUNICORN (GREEN UNICORN), UWSGI, MOD_WSGI, CHERRYPY.
—> It can only handle one asynchronous events per applications.

iii. asgi.py :
—> ASGI allows multiple, asynchronous events per application.
—> ASGI supports both sync and async apps.
—> It is built as a successor to the Web Server Gateway Interface (WSGI)

iv. manage.py :
—> It is Django's command-line utility for administrative tasks.
—> It is used for many different tasksment, like :

- **Running the development server : python manage.py runserver**
- **Doing database migrations : python manage.py makemigrations and then python manage.py migrate**
- **Running tests**
- **Running management project-specific commands for your Django project.**

python -m django startapp <App_Name>
python manage.py runserver

DJANGO WEEK 3

9. What is settings.py in django and its use?
—> The settings.py file is typically used to store configuration information in Django.
—> It may contain the site URL, paths to various directories and other values that the executable code may use. By changing and supplementing this file, the developers configure Django Projects.

BASE_DIR : The paths we define in the project are all relative to BASE_DIR

DEBUG : In Development Error is very obvious to occur. Django provides an inbuilt Debugger which makes the developer's life very easy. It's default value is true only in Development Phase. In production, DEBUG = False is recommended

ALLOWED_HOST : It is list having addresses of all domains which can run your Django Project.
—> When DEBUG = True, it can be an empty list because by default it is 127.0.0.1 or localhost
—> When DEBUG = False, it can not be an empty list. We have to give hosts name in list.

INSTALLED_APPS : In this section, we mention all apps that will be used in our Django Project.

DATABASES : Django officially supports the following databases :

- **PostgreSQL**
- **MariaDB**
- **MySQL**
- **Oracle**
- **SQLite <--- By Default**
- **NOTE : Before using PostgreSQL we have to install psycopg2**

```
DATABASES = {
        'default' :    {
                'ENGINE' : 'django.db.backends.postgresql',
                'NAME' : YOUR_DB_NAME,
                'USER' : USERNAME,
                'PASSWORD' : PASSWORD_FOR_DB,
                'HOST' : 'localhost'
            }
    }
```

URL Variables : They are relative to BASE_DIR. These variables are used to store files either media or static.

NOTE : Make static and media folders in Parent Directory.

MEDIA_URL : It is relative path to BASE_DIR. This variable is used to store the media files.
MEDIA_URL = '/media/'

DJANGO WEEK 3

STATIC_URL : It is relative path to the BASE_DIR. This variable is used to store the static files.
STATIC_URL = '/static/'

ROOT Variables : ROOT Variables are absolute paths. These variables are used to retrieve files either media or static.

MEDIA_ROOT : It is an absolute path. This variable is used to retrieve the media files.
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

STATIC_ROOT : It is an absolute path. This variable is used to retrieve the static files.
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
NOTE : All variable names in Django settings are in CAPITAL.

What is views.py in Django?
—> Django views are python functions that takes http requests and returns http response, like HTML document
—> A web page that uses Django is full of views with different tasks and missions.
—> It contains the logic that user sees in the web app.

10. What is urls.py and its use?
—> It is used to configure the URL patterns for your application. It is where you define the different URLs that your application will respond to and the views that will be called when those URIs are accessed.
—> The include() function is used to include the URL patterns from another 'urls.py' file and to specify the application namespace. This allows you to split the URL patterns for a large application into multiple files and to distinguish between different instances of the same application.

11. What is models.py and its use?
—> It is one of the most important concept of django framework.
—> It allows us to completely define the database of our web applications allowing us to :

- Declare tables and fields of our database.
- Define all meta-data of our Database.
- Decide the behaviour of each field.

—> We run the command **python manage.py makemigrations** this command create a new 'migration' of our models.py file. Actually it scans and compares the models.py version to the version currently contained in the migration files and write a new set of migrations.
—> Run the command **python manage.py migrate** this command will synchronizes the database state with the current set of models and migrations.

12. What is admin.py in Django and its use?
—> The admin.py file in Django is used to configure the admin interface for your application. It is where you define the 'ModelAdmin' classes for your models, which specify how the models should be displayed and edited in the admin interface. You can also use the admin.py file to customize the admin interface in various ways.

DJANGO WEEK 3

13. What is configure many DB in single Django Project?

—> To configure Multiple Database in a single Django Project, you need to follow these steps :

1. Configure the databases in your ‘settings.py’ file. Here’s an Example of how you can configure two databases, ’database1’ and ‘database2’ :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'database1',
        'USER': 'user1',
        'PASSWORD': 'password1',
        'HOST': 'localhost',
        'PORT': '',
    },
    'database2': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'database2',
        'USER': 'user2',
        'PASSWORD': 'password2',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

2. Create a router to determine which database to use for each model. Here’s an example of a router that uses ‘database1’ for the ‘Model1’ model and ‘database2’ for the ‘model2’ model:

```
class MyRouter:
    def db_for_read(self, model, **hints):
        if model == Model1:
            return 'database1'
        if model == Model2:
            return 'database2'
        return None

    def db_for_write(self, model, **hints):
        if model == Model1:
            return 'database1'
        if model == Model2:
            return 'database2'
        return None

    def allow_relation(self, obj1, obj2, **hints):
        if obj1._state.db in ('database1', 'database2') and obj2._state.db in ('database1', 'database2'):
            return True
```

```
        return None

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if db in ('database1', 'database2'):
            return True
        return None
```

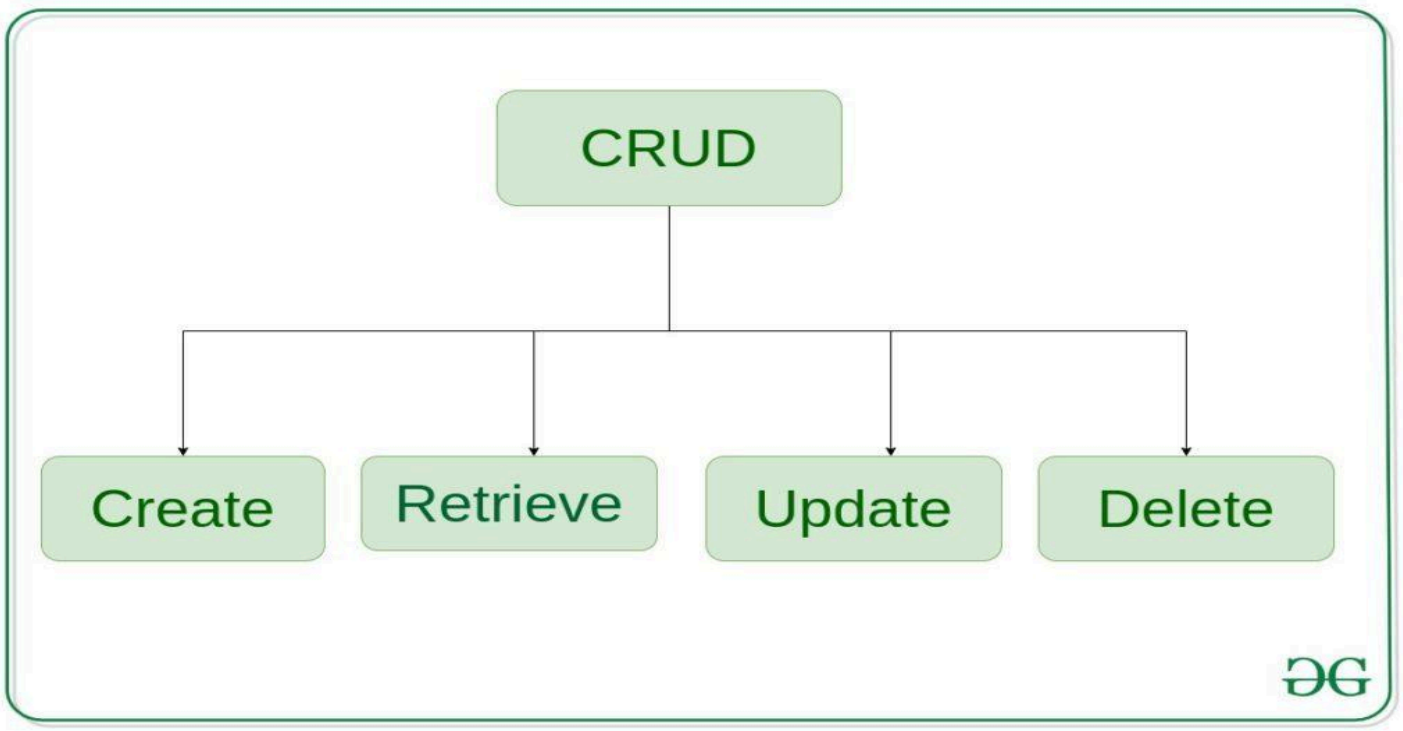
3. Add the router to the ‘DATABASE_ROUTERS’ setting.
DATABASE_ROUTERS = ['path.to.MyRouter']

4. Now you can use the ‘using’ method to specify which database to use for a query :
Model1.objects.using('database1').all()
Model2.objects.using('database2').all()

5. When running migrations, you can specify which database to use with the ‘--database’ option :
python manage.py migrate --database=database1
python manage.py migrate --database=database2

—> These are the steps to configure multiple database into single Django Project.

14. What is CRUD Operations in Django?
—> Django is based on MVT (Model, View and Template) architecture and revolves around CRUD (Create, Retrieve, Update, Delete) operations.
—> CRUD can be best explained as an approach to building a Django web application.
—> In general, CRUD means performing Create, Retrieve, Update and Delete operations on a table in a database.



DJANGO WEEK 3

Create - Create or Add new entries in a table in the database.
Retrieve - Read, retrieve, search or view existing entries as a list or retrieve a particular entry in detail.
Update - Update or edit existing entries in a table in the database.
Delete - Delete, deactivate or remove existing entries in a table in the database.
—> Check VS Code for Example…