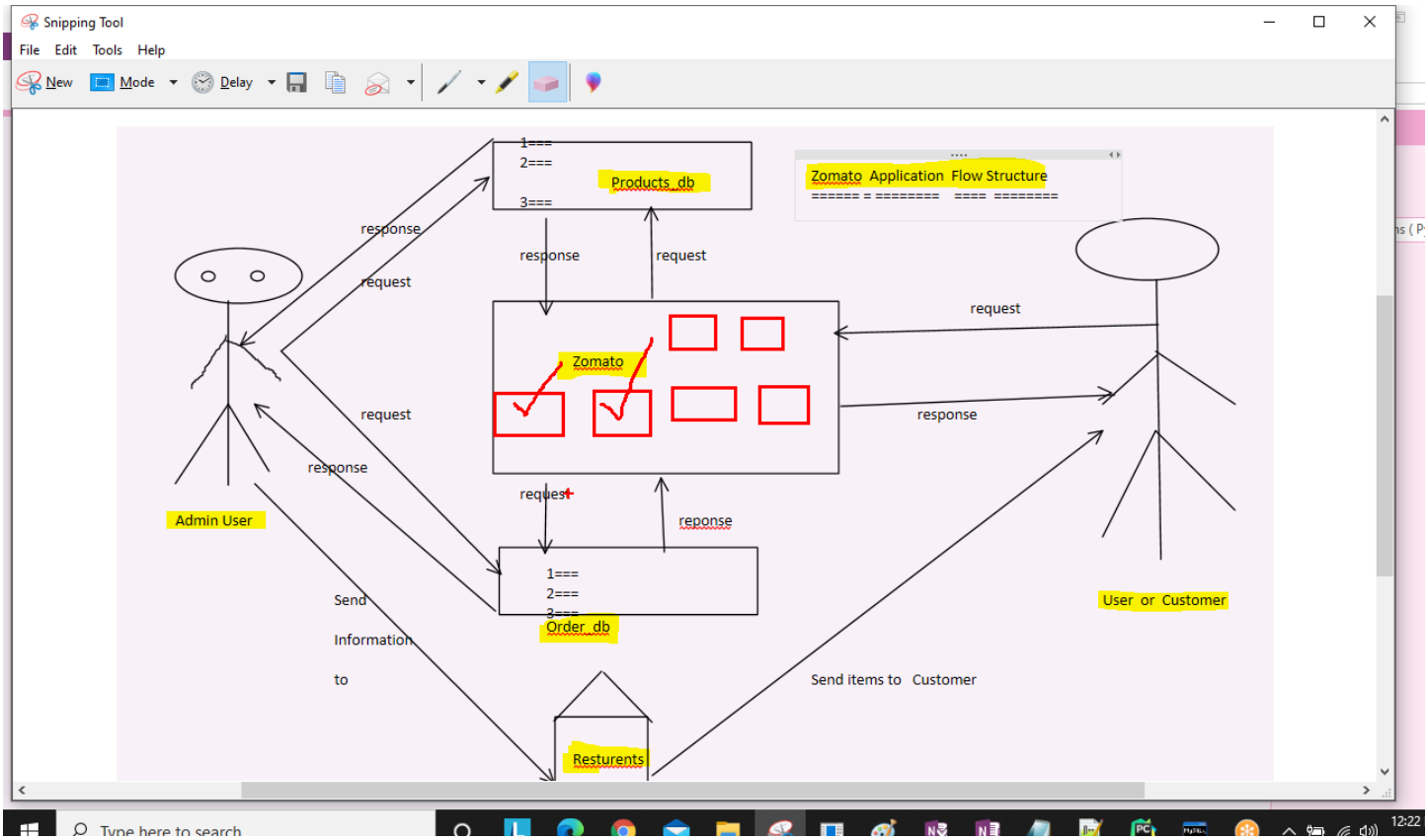
Introduction to Database Communication using Django

- Database is an important part of every website.
- Generally every website must required database to store the data.
- Database provides communication between Users and Admin of the organization.
- Any website which is using the database(s) then that website is called Database Driven Website.
- Django is wel-suited to create Database Driven Websites.
- Generally, both admin people and users use the database , like below



Zomato App Working flow Structure

- Admin or manager of the website will store the details of the products in the database (base_db) and display on the browser.
- Users or Customers will check required product on the browser, if the user finds the required product then He/She will place an order, this product order data will save in the another or same database.
- Admin or manager will keep on check the database (order table) for new orders from the Users or Customers..
- If Admin or manager finds any new order then they will inform to corresponding Restaurent people to deliver the products to the customers.
- Here, Admin uses base_db to store the products details and order_db used to check the User orders.

- Users use only order_db to store their orders.
- That means both admin and users are using the database.

Different Databases:

- Django supports all types of databases to use in the website,
- Django supports all major databases like Oracle DB , MySQL DB , SQL Server DB , PostGreeSQL DB and also SQLite3 (It is default database in django)
- We configure database details in settings.py file

Configuring Database:

- We have to configure the database details in settings.py file. By default it contains sqlite3 database settings in settings.py file.
- We are not required to download and install sqlite3 database.
- When ever we are excuting the runserver command then Django will download and maintain sqlite3 db in project root folders's location.

For example,

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': BASE_DIR / 'db.sqlite3',
   }
}
```

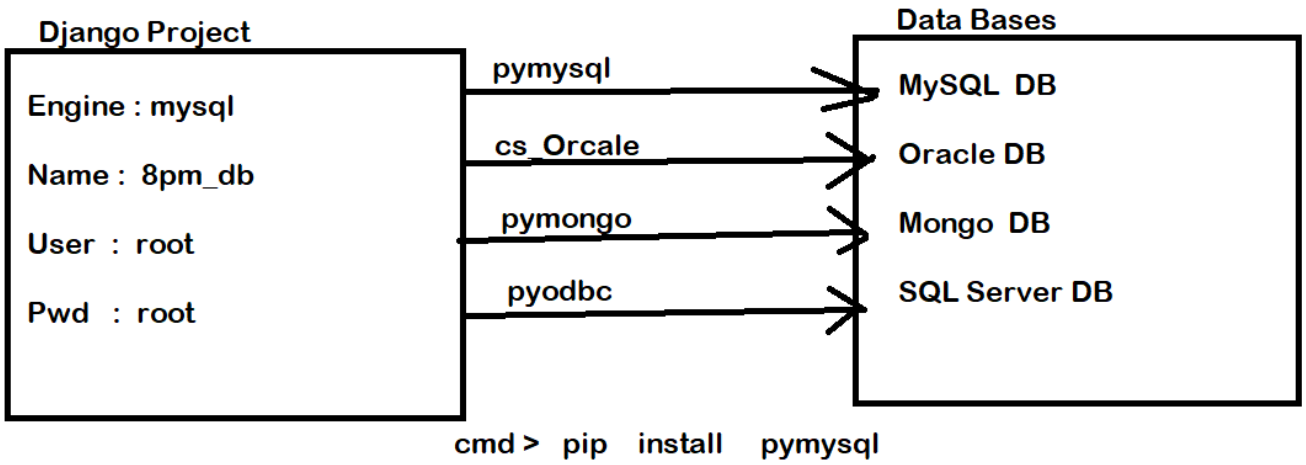
Note: If you want to connect to any other major database like MySQL DB then use database settings like below

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.mysql',
      'NAME': '8pm_db',
      'USER' : 'root',
      'PASSWORD' : 'root',
   }
}
```

Connecting to MySQL DB :

If we want use or connect any database with Python or Django then we have to install the Corresponding module by using PIP command like bellow

```
C:\Users\Lenovo> pip  install  pymysql
```



Create a new Project to connect to the MySQL

Step 1: Create a Project name like ModelProject

Step 2: Create a Application name like ModelApp

Step 3: Open mysql command prompt and create database name with 8pm_db

mysql > create database 8pm_db;

Step 4: Open settings.py file and

1) Add our application name inside `INSTALLED_APPS` section

For example,

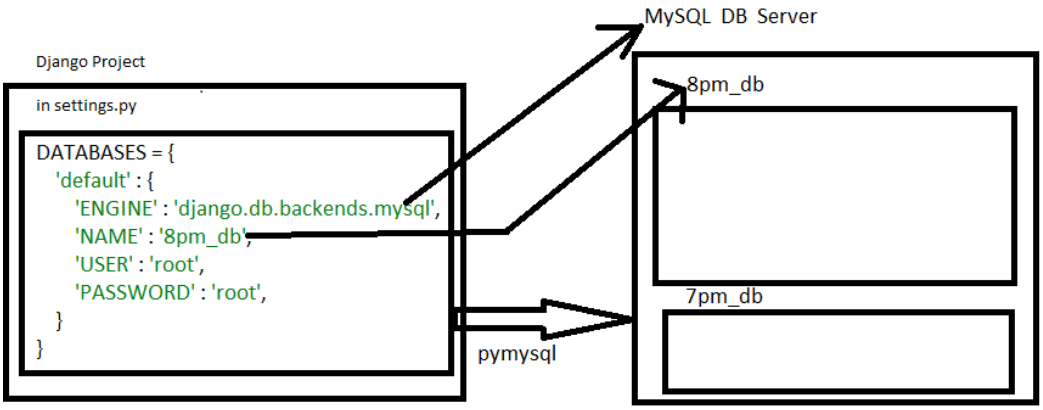
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

   'ModelApp',  #  our  appName
]
```

2) Configure the mysql db details inside DATABASE section.

For example

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.mysql',
      'NAME': '8pm_db',
      'USER' : 'root',
      'PASSWORD' : 'root',
   }
}
```

When ever you specify the database details in Python or Django, It creates a connection like bellow.



Django to MySQL DB Connection Setup

1. pymysql module is acting like a connector module between django and MySQL DB.
2. It uses username and password values and making connection
3. Using NAME key we can represent which db name we want to connect in MySQL DB.
4. If db name not avialable then it throws UnknownDatabase Exception.
5. ENGINE key represents to django that connect to MySQL DB

Testing Connection :

- If we want to know whether we specified proper database details or not in the settings.py file then we can execute the connection by using cursor().
- If cursor() is returns any Error that means we did not specify the database details properly in settings.py file.
- If cursor() did not return any Error that means we have given all database details properly in settings.py file.

Note: The connection is availble in django.db, so we have to import the "connection" object and execute the connection by using cursor()

Now we will goto python shell to verify the connection object , click on terminal, run the following command

E:\django8pm\modelpro> python manage.py shell

```
>>> from django.db import connection
>>> c = connection.cursor()
>>>
```

NOTE: Here cursor() did not return any Error that means we have given all database details properly in settings.py file.

If connection is created properly then we can create a model in models.py file.