Django

NARAYANA SIR

04/06/18

Web frame work:- It is a software which is used to develop Bros dynamic web applications. it is a server side application which provides the rapid development of dynamic web applications.

- It is a collection of classes (& API (Application Prog. Interface) which is a predefined code and which is used to create stand alone applications.

* **different types web frame works:-**

Language	Web frame works
Java	hibernet, spring, struts
C++	CPPCMS, POCO
Java Script	NodeJS, Angular JS
Php	Yii, Zend
Python	django, flask, Pyramid, web app engine, pylons, pyjs, cubicpy

History of Django framework:-

- Initially Django framework started as a intranet project in lorenG-gemal-valls news paper in 2003.
- we can also say that it is started in a small news room.
- Andriam & Eyman are the web developers at lorenG general of news paper, They are responsible, to create and manage multiple web sites to meets the dead lines of the web sites as per requirement.
- They factor out all common features from all the web sites and they framed a frame work with this features. after compiling & debugging

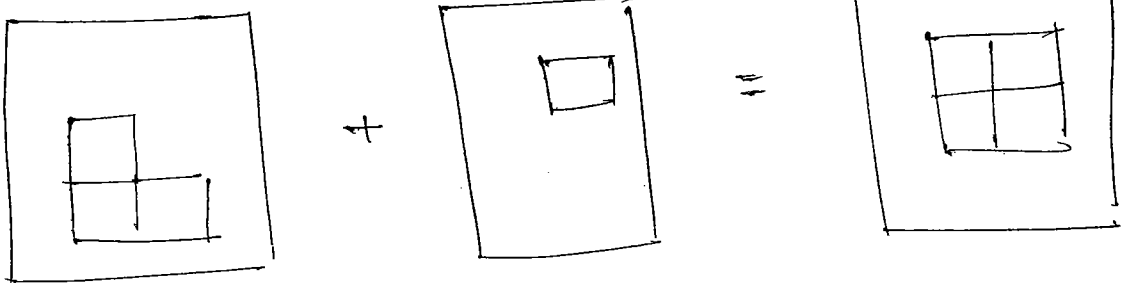
the features in multiple types.
- The developer like music and guitarist name called 'Django reinhard' thats why they named the frame work as django frame work.
- In 2005 They tested multiple web site by using Django-framework and all website have given a netured results. (good)
thats why They decided to release Django frame work as 'open and resource'.
- In 2008 They formed a seperate organization (DSF) django software foundation which is a independent & non profit organization.
- They released Django frame work in 2008 as a first version.
- now the latest version is 2.0.

Advantages of Django framework :-

① Django framework is a predefined (or) has a a predefined modules (or) packages (or) libraries that's why almost all frame work is predefined component.

② as a user we add very less code to the existing frame work to create our own websites as per our requirement.

(1) Django (75%) + User code (25%) = Final web app



③ The advantages are, - It takes less development time
- less developement cost
- less maintainance time.
- It gives more productivity (Nothing but achieving more results)

with in less time and less cost.

Features of Django framework:-

(1) very fast
(2) scalable (can dev. small & large websites) medium, high flow
(3) versatile (all types of web sites).
(4) very secured
(5) easy to maintain
(6) Portable (It can run on any platform)
(7) Brings tons of packages.
(8) open and free source. (No license required)
(9) default data bases (SQL) sqlite3 → At the time of runserver
all databases (oracle, mysql, sqlite3, and Postgresql)

Python Installation:-

→ www. python. org.
→ click on download option
→ click on python latest version 3.6
→ go to downloads folder and click on python software.
→ check add path (we can also add manually).
and click on install now.
→ It Installs completely.

25/06/18

PIP( ):- (python package installer)

this command is used to install any package regarding python.

- It is an alternative for easy install command
- easy install was install 2004, PIP was released in 2008
- these two commands are automatically installed when we use python software in,

C: \Users \Narayana \App Data \ local \ programs \ Python \ Python 36-32 \ scripts

C: \ Python 27 \ scripts.

Syntax:- pip install <pkg_name>

eg:- pip install django
easy_install django

Django installation:- we need to install python software to install Django.
- we use pip command to install Django.
- goto command prompt and Run the following command.

* >pip install django.

- The above command will install the latest version of Django.
- If we need to install any previous or any old versions then we add == version no. (ex 1.9)

pip install django==1.9 ✓

- whenever we install django Then it creates a new command line utility.

python36-32/scripts folder. ((django admin) ↓ file name)

- The file name used to perform administrative operations of a project.

Django project:- A project is a collection of applications with their configurations.
- A project should contain atleast one application.
- we use django-admin file to create Django project.
- we use start project command to create a django project.

Step (1):- create new folder to store the project.

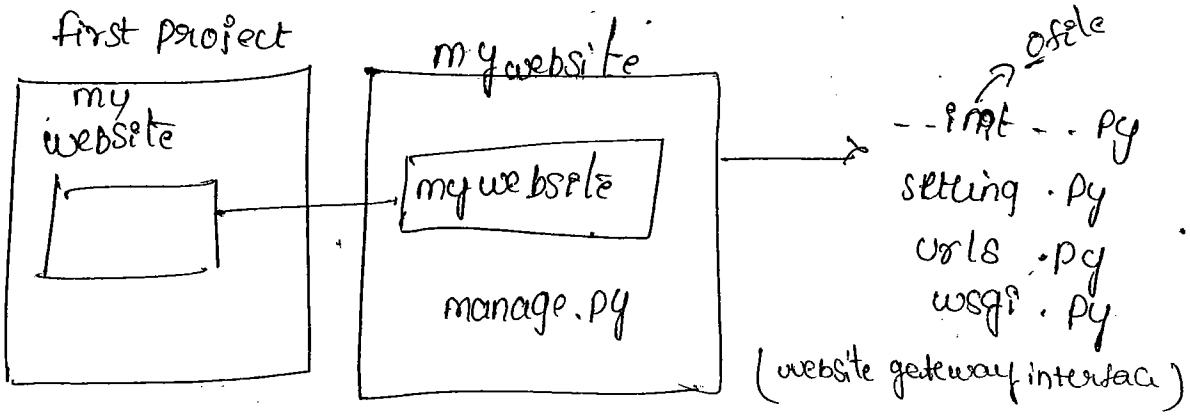
E:\first project
folder.

Step② :- open first project folder and type cmd in path. to open command prompt.

E:\first project>

Step③:- Create a project with name 'mywebsite'.

E:\first project > django-admin start project mywebsite



Step④:-

we need to used manage file to run the server, but manage.py file is available in mywebsite folder. So to change the path to my website.

E:\first project > cd my website

Step⑤:- now we run runserver command to start the server.

E:\first project\mywebsite > python manage.py runserver

Step6 :- now copy the IP address http://127.0.0.1:8000/

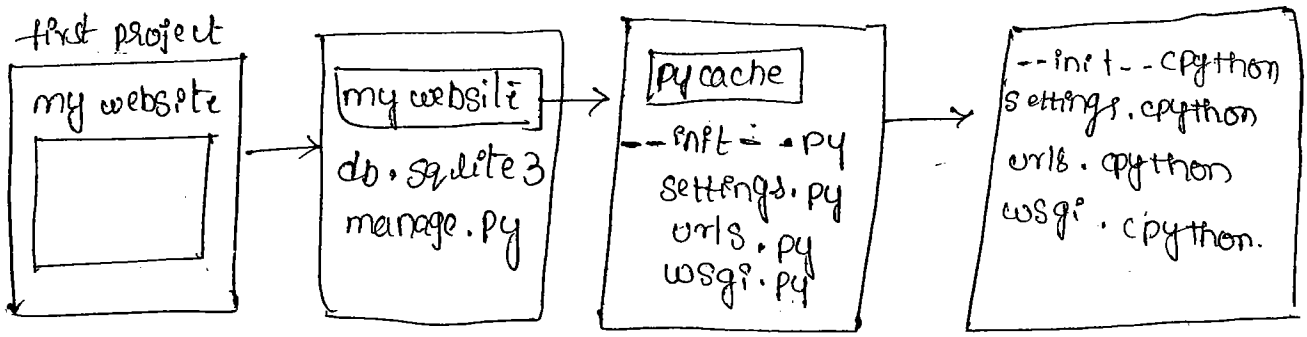
Step7:- open any browser and paste the IP address.

It worked!

Congratulations on your first django-powered page

6/06/18

* when ever we run the server then django will install sqlite3 database in my website folder and also it compiles all project level files and create a new folder __pycache__



Applications:-

→ An application is a part of Django project. each application has it's own task (or) Aim.

→ we can create multiple applications in a single project i.e depending on our requirement.

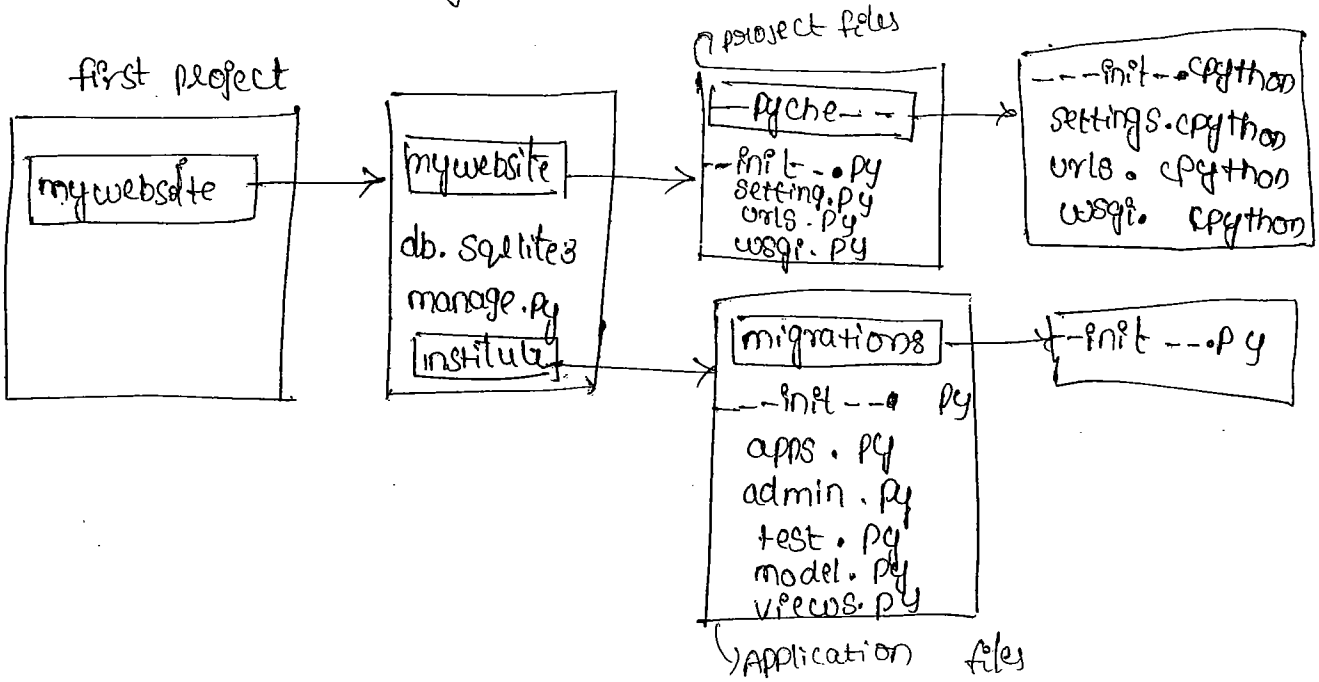
→ If we need a project to aim only one task then we create only one application; If we need a project to do different number of unrelated tasks then we create multiple applications.

→ We use 'start app' command to create a application, this 'startapp' command is a subcommand of manage.py file, thats why we goto the location wherever manage.py file located and in that location we create our applications.

Step(1):- go to my website folder and create application.

E:\first project-\my website > python manage.py startapp institute

→ now we can see institute folder in mywebsite folder whereever mange.py file folder there.



→ open our project in Pycharm

file ---→ open ---→ choose the project

→ open institute / views.py and write the below code.

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

def index(request):
    x = 'welcome to durgasoft'
        <h1>                </h1>
    return HttpResponse(x).
```

→ open mywebsite / urls.py file and write the below code.

```
from django.conf.urls import url
from django.contrib import admin
from institute import views.
```

```
urlpatterns = [
    urls (r'^admin/', admin.site.urls),
    urls (r'welcome/', views.index),
]
```

→ go to the location where ever manage.python is available & run the server.

E:\firstproject\mywebsite > python manage.py runserver

↓

→ copy the IPA address and paste in any browser.

→ 127.0.0.1:8000

page not found (404)

1. ^admin/
2. welcome/

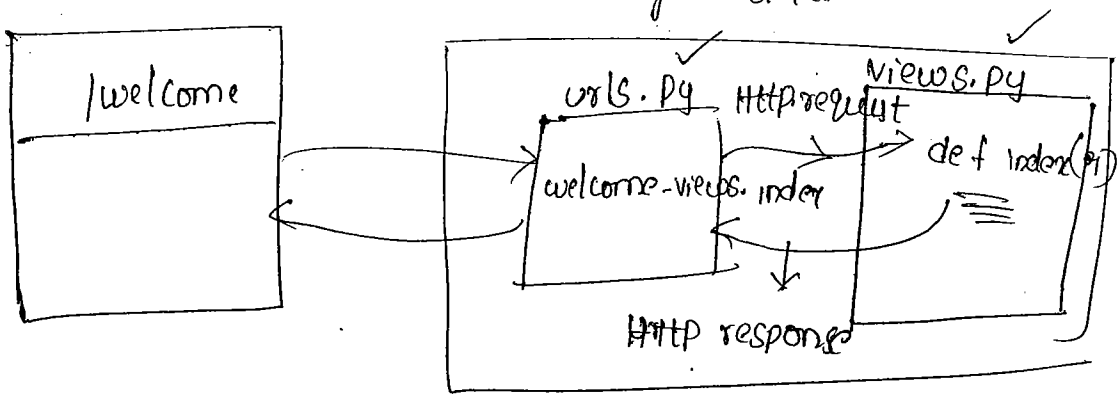
→ we need to specify either admin (or) welcome in the url to get proper response.

→ 127.0.0.1:8000/welcome/ → keyword.

Welcome to durgaSoft

^ → start of the word
/ → end of the word

r — raw string.



Views :- A view is nothing but a Python function which takes HTTP request from user and gives http response to the user.

→ Always views interact with models to store (or) retrieve the data from the database.

→ views also interact with Templates to manage the presentation code. (HTML)

→ views are responsible to perform some arbitary logic as per the requirement.

→ we also import views in urls.py file to give the mapping b/w the key words and the corresponding function (URL Coflicts) which is defined in the view.py file.

Ex :- 2

home → welcome to DS
Courses → we provide all S/w courses
Services → we provide excellent services
Python → we start Python class every week
Django → we start django class for every two weeks.

8/06/18

③ current date and time. when user enters time keyword.

Step①:- create a folder name - date_folder

E:\datefolder

Step②:- open datefolder and type cmd on path to open command prompt with current location.

E:\date_folder>

Step③:- create a project with name datepro.

E:\datefolder>django-admin startproject datepro

Step④:- change the path to the location wherever manage.py file.

E:\Datefolder>cd datepro

↓path

E:\datefolder\datepro>

Step⑤:- create an application with name dateapp

E:\datefolder\datepro> python manage.py startapp dateapp
file command pr name

Step⑥:- open over project in pycharm.

Step7:- go to dateapp/views.py file and write the below code.

```
from django.http import HttpResponse
import datetime

def current_time(request):
    n=datetime.datetime.now()
    m = '<h3> the current date and time is %s </h3>' %n
    return HttpResponse(m)
```

Steps :- go to datapro / url file → to map key, urls / views

```
from django.conf.urls import urls
from django.contrib import admin
from dateapp import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^time/', views.current_time)
```

Step9 :- goto commandprompt and run the server

E:\datefolder\datapro > python manage.py runserver.

Step10 :- copy the IP address and open any browser and paste IP address along with ~~time~~

The current date and time is 2018-06-08 09:47:47.9007

Working with models :- till now we developed some projects around views.py file and urls.py file.

→ Generally all ~~databases~~ websites interact with database to manage the information about the products / services and so on...

→ The website which is interacting with database is called 'Database driven website'

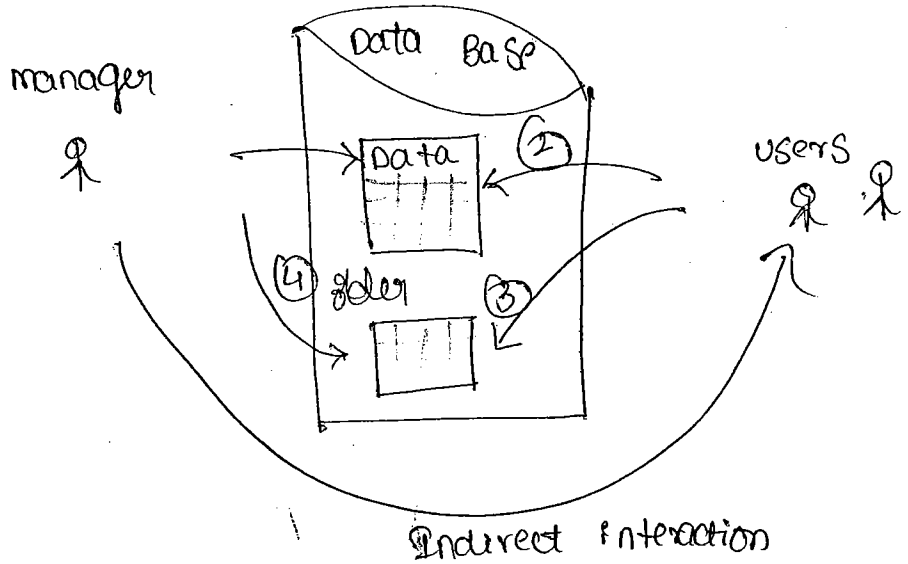
→ Django framework is well suited to develop data base driven websites.

→ Generally the managers of website and customers (or) users interact with database.

→ The managers will insert all update the data in the data base.

→ The users will see the data and they place the order as per his requirement.

→ The managers will see the order and they deliver the product to the customer.



Indirect interaction

→ Django supports all data bases but officially supports Oracle, mysql, sql server, and sqllite 3.

10-06-18

Data-base Configuration:-

- Django support all databases but it support officially ORACLE, MYSQL, POSTGRESQL and SQLLITE 3.
- we give data_base configuration in setting.py file.
- the default database configuration in setting.py file
- the default database 'sqlite3' required only ENGINE and NAE.

— Here, ENGINE name is server name and NAME means database name.

```
DATABASES = {
    'default' : {
        'ENGINE' : 'django.db.backends.sqlite3',
        'NAME' : os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

⇒ If you want configure any other databases then we need some other parameters, like

* ENGINE
* USER
* NAME
* PASSWORD
* HOST
* PORT

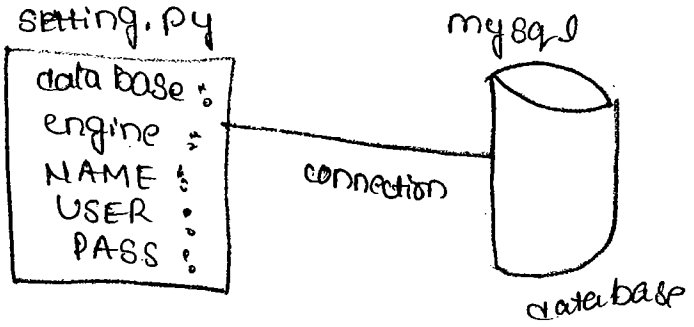
— Generally HOST & PORT are optional for all databases when we work with local projects.

✓ If we need to configure mysql then we have to do the following changes in DATABASES option in settings.py file

```
DATABASE = {
    'default : {
        'ENGINE' : 'django.db.backend.mysql',
        'NAME' : 'mydb',
        'USER' : 'root'
        PASSWORD : 'root'
        }
}
```

— we can crosscheck whether we have given proper database details or not by using 'shell'.

→ generally, when ever we give database configuration in setting.py file then automatically a connection will be created b/w our django project and specified database.



- now, we can test the connection executing this connection with cursor().

```
E:\first project\mywebsite> python manage.py shell
>>> from django.db import connection
>>> c = connection.cursor()
>>>
```

→ Here this connection as not returned any errors. i.e we have given Proper database configurations

→ now we can create modes in this database.

Creating model :-

Def :- A Model is a Python class which contains a model name, field names and field types.

→ all user defined modes are sub modes of django.db.models.Model.

→ we goto models.py to create a model.

→ How the model names become table name, field name becomes column names and field types becomes data-types in the database.

modelname → table name
field name → column name
field type → data type

models.py

```
from django.db import models

class Employee (models.Model):
    eid = models.integerfield()
    ename = models.characterfield(max_length = 20)
    sal = models.integerfield()
    loc = models.characterfeeld(max_length=20)
```

Activating a model:- we have to activate our models we need to specify our application name and INSTALLED APP in settings.py file.

setting.py.

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

↓ application name

→ Every django project has 6 default applications, and INSTALL-APPS, Thus are default applications run automatically when we run the server.

→ If we need to run application also automatically Then we have to specify our application name and INSTALLED APP.

Make Migrations:-

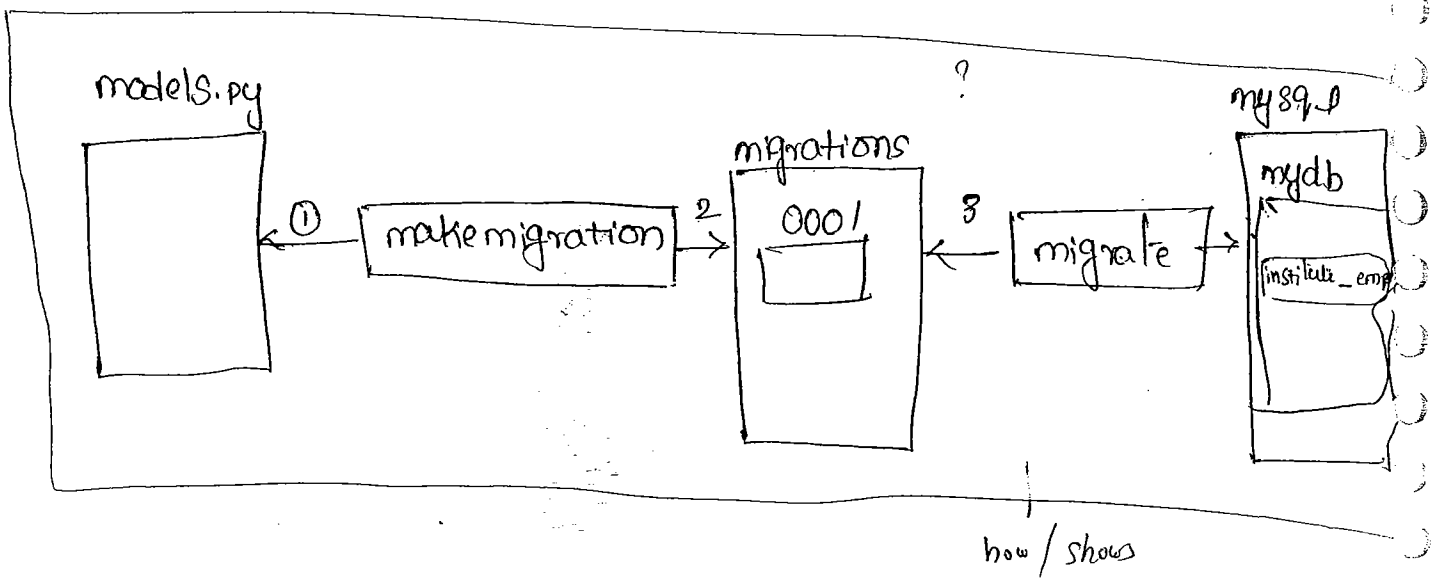
→ we need to Run make migration command to capature the recent migration's from the models files and creates a new file with recent migration's in migrations folder.

→ E:\ FirstProject \ mywebsite > python manage.py makemigrations

→ This command will not create the table in the database.

→ It just creates a new file in the migrations folder.

file name : 0001_initial.py
↓ column names to all field
: create Employee model



→ this command creates multiple new python files in migrations folder for multiple recent migrations.

→ if we run make migrations without having any changes in model file then it display,

> No changes are detected

Sqlmigrate :-

→ This command is used to see the SQL code of a specific file.

> syntax :- manage.py sqlmigrate institute 0001

> E:\firstproject\mywebsite> python manage.py sqlmigrate institute 0001

```
create table institute_Employee (
    'id'    integer
    'eid'   integer
    'ename' varchar(20)
    'sal'   integer
    'loc'   varchar(20)
);
```

field name — datatypes

→ In every model has one default field Id which is a primary key and Autoincrement.

→ generally we don't give the data to the default id column.

→ now we need to create table in db.

→ we have 2 ways to create a table in the db.

(1) Through migrate command

(2) copy sql code and paste in the server and run the code.

① Through migrate command :-

E:\firstproject\mywebsite > python manage.py migrate.

→ this command creates somany tables regarding Authentications, Admin, and permissions along with our table name.

institute_employee

→ now open mysql command line client

Enter password : root

mysql > show databases; (It will display all databases including mydb database)

mysql > use mydb;

Database changed

mysql > show tables;

Table_in_mydb
auth_group
auth_user
auth_permission
institute_employee

mysql> desc institute_employee; ✓

Table name

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment.
eid	int(11)	NO			
ename	varchar(20)	NO			
Sal	int(11)	NO			
loc	varchar(20)	NO			

Table name :-

It is the combination of application name and lower case of model made with underscore B/w this two.

institute_employee

11-06-18

→ we can populate (or) insert (or) give the data to the table in different ways.

(1) we can given through python shell

(2) we can give through Browser

(3) we can give through Admin

(1) Inserting data through Python shell :-

→ we can insert the data through *save methode (or) *create methode

(i) Save methode :- In this process to take data into a new object and then to save that object with save methode

→ Go to python shell

> python manage.py shell

>>

→ we have to import the employee model to the shell.

>>> from institute.models import Employee.

↓next

>>> a = Employee (eid = 101, ename = 'Nani', 'sal' = 10000, loc = 'Hyd')

a.save()

>>> b = employee (eid = 102, ename = 'sai', 'sal' = 20000, loc = 'Ban')

>>> b.save()

→ now we can see the above two employees data in the data-base Table.

→ go to my sql and run select statement.

mysql > Select * from institute_employee

id	eid	ename	sal	loc
1	101	Nani	10000	Hyd
2	102	sai	2000	Ban

(ii) Create ( ) :- Here we can take data into a new object. we don't require to save the object, Bcoz create ( ) "automatically save the data in the data Base."

>>> c = Employee.objects.create (eid = 103, ename = 'Amit', sal = 30000, loc = 'pune')

>>> d = employee.objectcreate (eid=104, ename='Sunil', sal=30000, loc='pune')
mumbai

→ now Run Select statement in the database to see the complete data.

mysql> Select * from institute_Employee;

id	eid	ename	sal	loc
1	101	Nani	10000	Hyd
2	102	Sai	20000	Bang
3	103	Amit	30000	pune
4	104	Sunil	30000	pune mumbai

(2) Inserting the data TO the Server:-

mysql> insert into institute_Employee (eid, ename, sal, loca) values (105, 'sumit', 40000, 'chennai')

mysql> insert into institute_Employee (eid, ename, sal, loc) values (106, 'durga', 30000, 'hyd')

mysql> select * from institute_Employee;

id	eid	ename	sal	loc
1	101	Nani	10000	Hyd
2	102	Sai	20000	Ban
3	103	Amit	30000	Pune
4	104	Sunil	80000	mumbai
5	105	Sumit	40000	chennai
6	106	durga	30000	hyd

(3) Inserting data through Admin :-

→ We have to know how create superuser and password to login into admin side. We will discuss this way of inserting data after discussing Admin concept.

12/06/18

Performing CRUD operations (through pythonshell).

① Step1 :- Create a folder to store the project

```
E:\second folder
```

Step2 :- open and folder and type cmd in path to open command prompt

```
E:\ Secondfolder>
```

Step3 :- create a project with name 'crudpro'.

```
> E:\ secondfolder>django-admin startproject crudpro
```

Step4 :- change the path to the location coherever manage.py file is available

```
> E:\secondfolder>cd crudpro
```

```
> E:\secondfolder\crudpro>
```

Step5 :- create an application with name 'crudapp'

```
E:\secondfolder\crudpro> python manage.py startapp crudapp
```

Step6:- Open pycharm

Step7:- goto settings.py file to

settings.py

```
DATABASE = {
    'default': {
        'ENGINE' : 'django.db.backend.mysql',
        'NAME' : 'mydb',
        'USER' : 'root',
        'PASSWORD' : 'root',
        }
    }
```

Step8:- goto models.py file to create a customer model.

```
from django.db import model is

class customer (models.model):
    cid = models.integerfield ( )
    cname = models.charfield (max_length = 20)
    sal = models.integerfield ( )
    email = models.Emailfield (max_length = 50)

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

Step9 :-

```
INSTALLED_APPS = [
    —
    —
    —
    —
    —
    —
    'crudapp',
```

Step10 :- Run makemigrations to capture recent migrations (changes) from models.py file and to create new python file in migrations folder with migrations (changes)

```
E:\second-folder\crudpro> python manage.py makemigrations
migrations for 'crudapp':
  0001_initial.py:
    . Create model customer
```

Step11 :- Run sql migrate command to see sql code of a specific file.

```
E:\second-folder\crudpro> python manage.py sqlmigrate crudapp 0001

Create table crudapp_customer(
```

Step12:- now we have to create the table

Step13:- we can copy the SQL code (CREATE TABLE statement) and run under mydb database in mysql.

Step14:- open mysql and activate mydb Database.

```
Enter 'password' : root
mysql > use mydb ;
Database changed
```

Step15:-

Paste the sql code (create a table statement) and run the code

```
mysql > CREATE TABLE 'crudapp_customer' ('id' integer AUTO_
INCREMENT NOT NULL PRIMARY KEY, 'cid' integer NOT NULL,
'cname' varchar(20) NOT NULL, 'sal' integer NOT NULL,
'email' varchar(50) NOT NULL);
                    ↓ ENTER
Query ok, 0 rows affected (0.14 sec)
```

Step16 :- run show tables ; to see our table name

```
mysql > show tables ;
```

Tables_in_mydb
auth_group
crudapp_customer
institute_employee

Step17:- Run desc crudapp_customer;

```
mysql> desc crudapp_customer;
```

Field	Type	NULL	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
cid	int(11)	NO			
cname	varchar(20)	NO			
sal	int(11)	NO			
email	varchar(50)	NO			

5 rows in set (0.11 sec)

12/6/18

Step18:- go to python shell to perform 'crud' operations

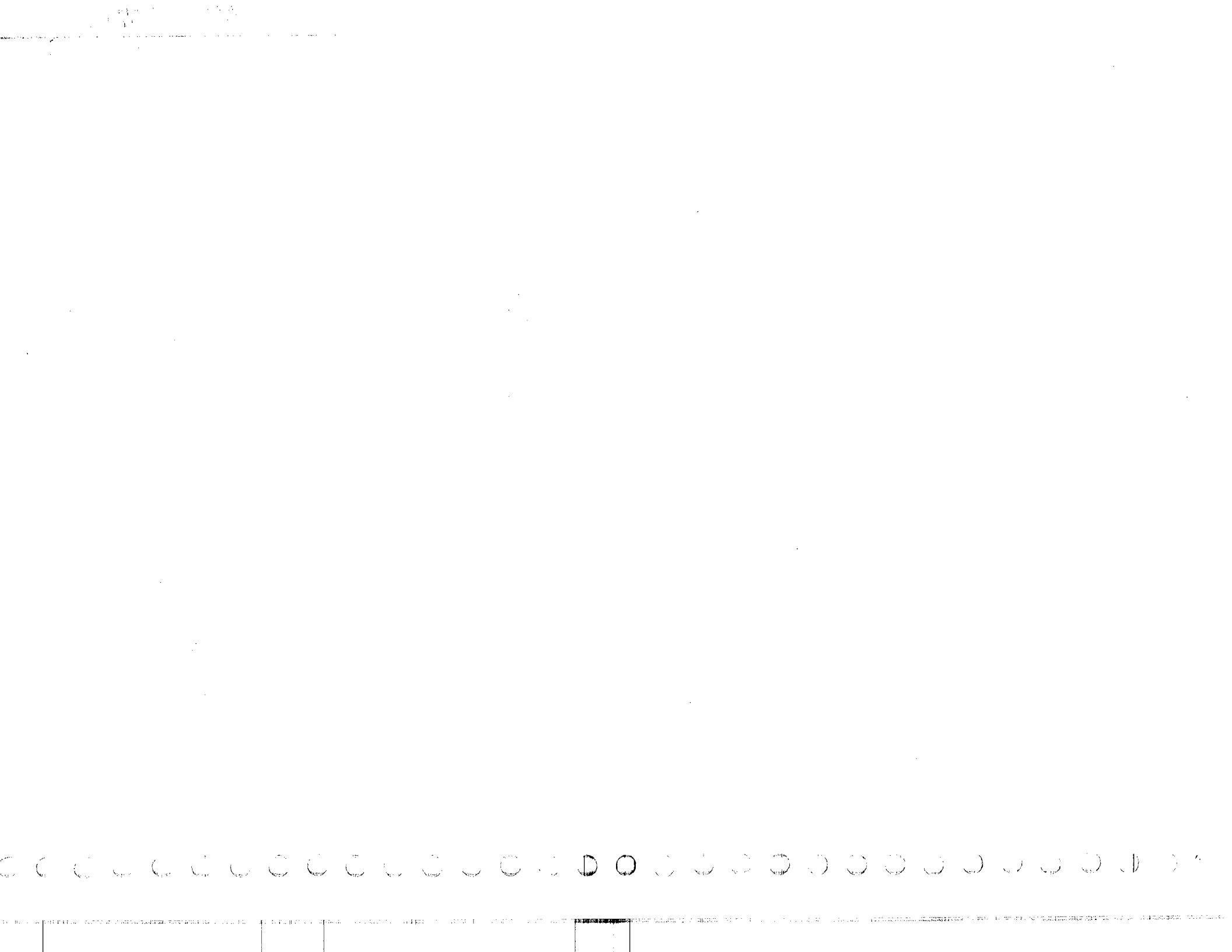
E:\second folder\crudpro> python manage.py shell

Step19:- Import customer modul to perform operations

>> from crudapp.modules import customer.

* Inserting the Data :-

a = customer



10/06/18

Admin Interface :-

→ Admin Interface is the essential part of any modern website.

→ we create username and p/w to the manager or admin (or) staff of the website so that they can login the website to insert or update or delete their data about Business.

→ Inserting, updating and deleting data in website is tedious (Boring) work. Bcoz there is no creativity in this operation.

→ But Django Brings an automatic admin interface to make the work very interest.

→ Django frame work was developed by in a small news room with a clear seperation b/w content publishers and public site.

→ Publishers are maybe managers (or) admin people (or) staff to publish the data of Business and customers (or) users will Interact with public site to see the content of the Business which is published by content publishers

→ we need to create username and p/w for content (manager) publishers.

step1:- create folder 'thirdfolder'

step2:- open thirdfolder and type cmd in path to open cmd

step3:-

E:\thirdfolder > django-admin startproject adminipro

Step4:- Change the path and create application with the name admin path.

path change

```
E:\thirdfolder> cd adminpro
E:\thirdfolder>adminpro> python manage.py startapp adminapp
```

Step5:- Create Superuser and password

```
E:\thirdfolder\adminpro> python manage.py createsuperuser
Username (leave blank to use 'narayana'): narayana
Email address: narayana@gmail.com
Password: 1234abcd
Password (again): 1234abcd.
Superuser created successfully.
```

Step6:- Run the server

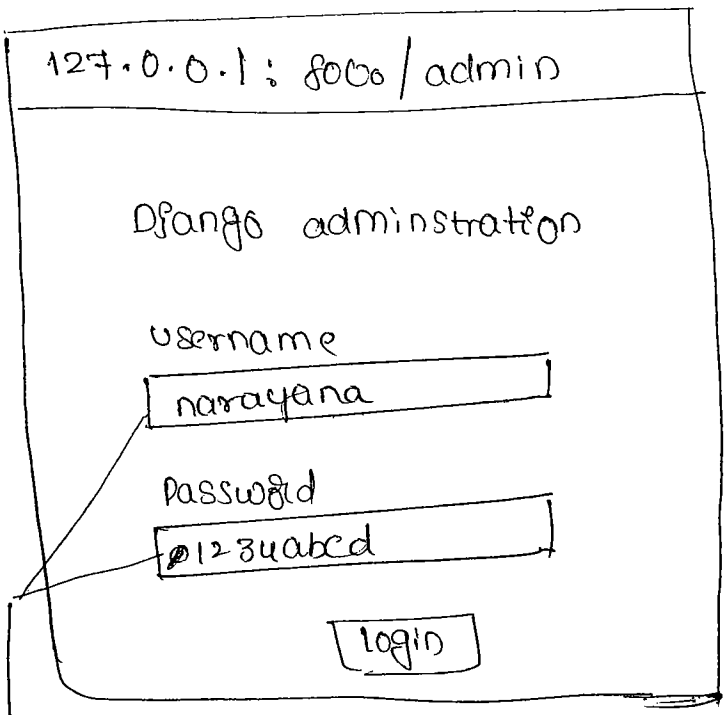
```
E:\thirdfolder\adminpro> python manage.py runser
```

copy the IP address (http://127.0.0.1:8000)

NOTE:-

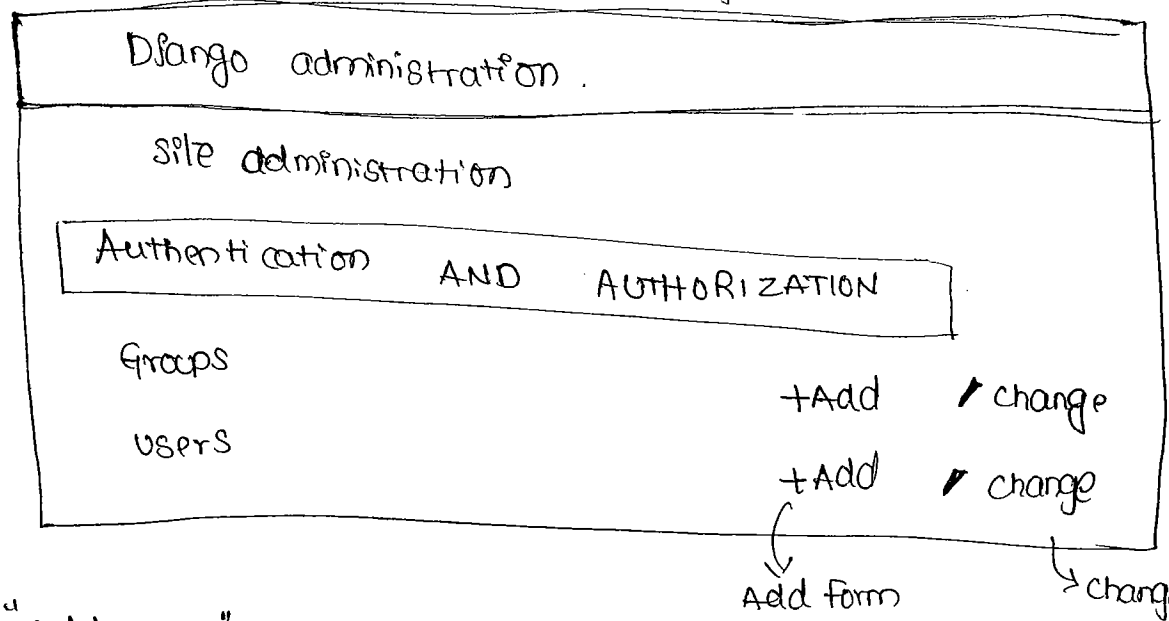
In django, if we get any error saying no such table & so on.... then we need to run migrate command to solve the error.

Step7:- open any Browser and paste any IP address along with /admin



Step 8:- Enter username and p/w and click login.

Then it opens 'admin home page'.



"Add form" is used to add new users (or) newdata to the site (or) table,

"change list" is to display all the users (or) Table data and also we can modify the users details (or) Table data.

15/06/18

Creating a user:- click on add form of users

Step 2:-

Add user

first enter username and password. then, you'll be able to edit more user options.

username : mounika

password : 1234abcd

Password confirmation : 1234abcd

Save & add another | Save and continue editing | Save.

→ if u click on save & add another user it will save and opens add user and form again to create new user.

→ if u click on save and continue editing then it will save the current user and it will ask for further information of current user.

like, first name, last name.

→ if we click on Save button it save the current user.

username : mounika

password : 1234abcd

Save

Permissions of users :- Django project supports 3 different "permissions" for every user.

1. Active :- By default it is selected i.e means all the user is an active member of the current website.

→ If u want to delete any user, simply unselect this active permission.

2. staff status :- If we select this permission for any user then he can login the website with his username and password.

→ The default this user has no permissions even though he can login the website.

→ if super user gives any permission then he can do that operations only. that means the permissions of this user is control by super user.

3. Superuser status :- if we select the permission for any user then he gets all permissions like super user.

→ By default he can not login the site with this permission, he should be given staff status permission also. so that he can login the site he has all permissions.

Loading Models into Admin side :-

step1 :- open 3rd (third) folder in pycharm.

step2 :- go to models.py file write the following

```
from django.db import models
class Employee (models.Model):
    eid = models.integerfield()
    ename = models.charfield (max_length=20)
    sal = models.Integerfield()
    email = models.emailfield()
    def_str_(self):
        return self.ename
```

Step3:
go to mysql and create database with name dbadmin.

```
mysql> create database dbadmin;
```

Step4:- go to setting.py file and activate our model and configure database

```
INSTALLED APP = [
    —
    —
    —
    —
    —
    'admin App',
    ]
```

```
DATABASE = [
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
        'NAME' : 'dbadmin',
        'USER' : 'root',
        'PASSWORD' : 'root',
    }
```

Step 5:- Run makemigrations commant (in cmd)

```
E:\Third-folder\adminpro> python manage.py makemigrations
```

Step6:- Run migrate command to create database tables.

```
E:\third folder\adminpro > python manage.py migrate
```

Step7:-

Run the server

```
E:\thirded folder\adminpro>python manage.py runserver
```

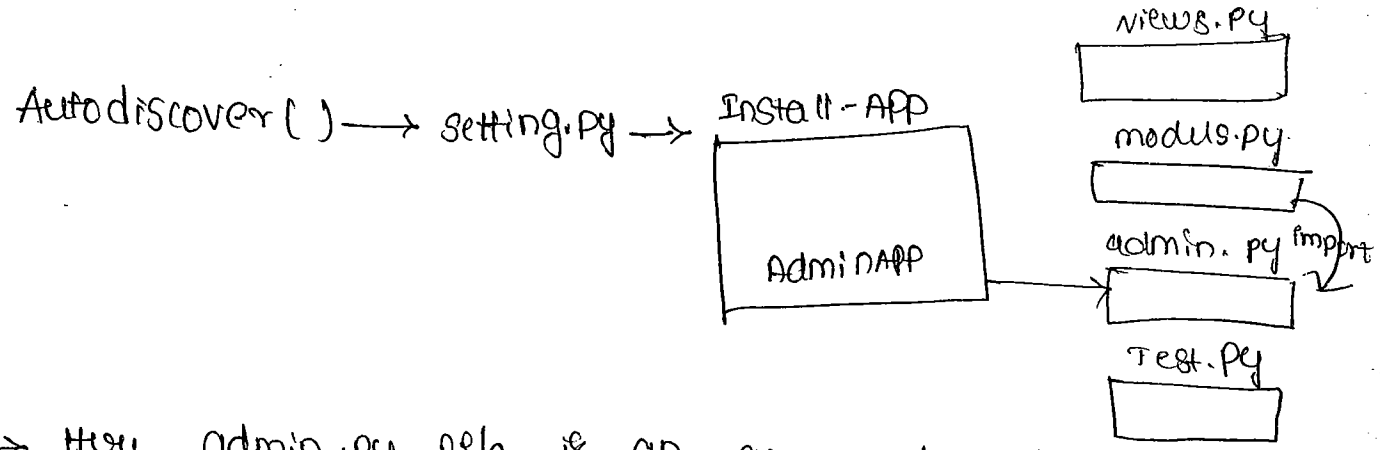
copy the IP address and paste any Brocoser along with `\admin` and login admin site with proper username and p/w.

Step8:-

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	/ Change
users	+ Add	/ Change

→ Here the modus are not loaded ~~Here~~ BCOZ the Admin site interacts the adminfile.

→ When ever we run the server then Automatically Autodiscover() coml go to the setting.py file then it go to the installed APP (adminapp) then it goes to admin.py file in adminapp application.



→ Here admin.py file is an an empty file by default Thats y nothing was admin site. we have to import the models from models.py file into adminfile and then we have to register with adminsite

Step9:-

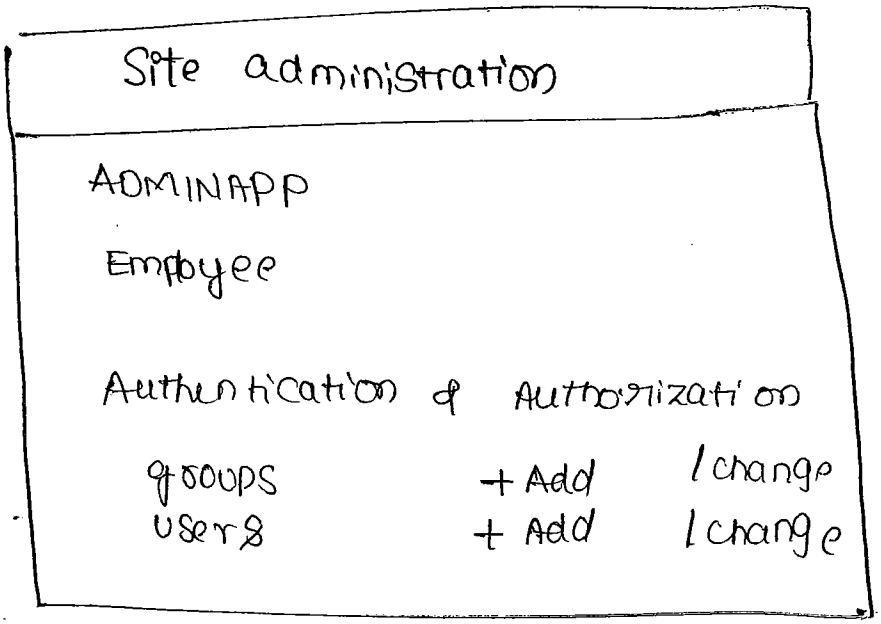
→ goto admin.py file:

admin.py

```
from django.contrib import admin
#importing model
from.models, import employee
# Register model
admin.site.register(Employee)
```

Step10:-

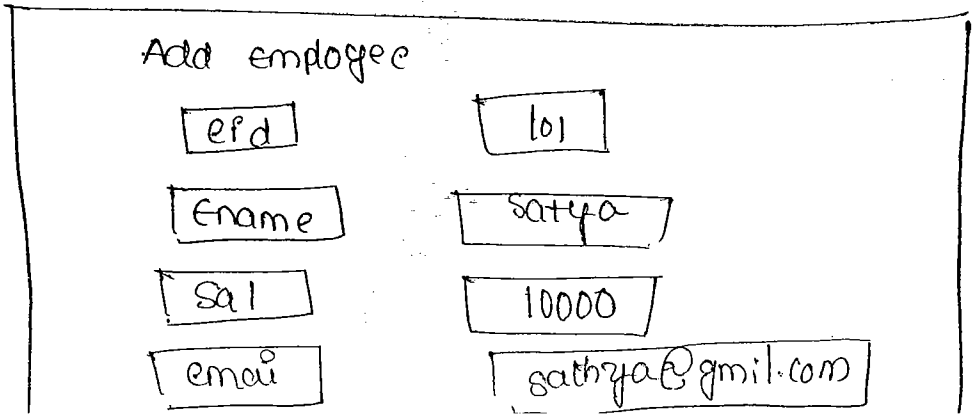
Now refresh the adminsite



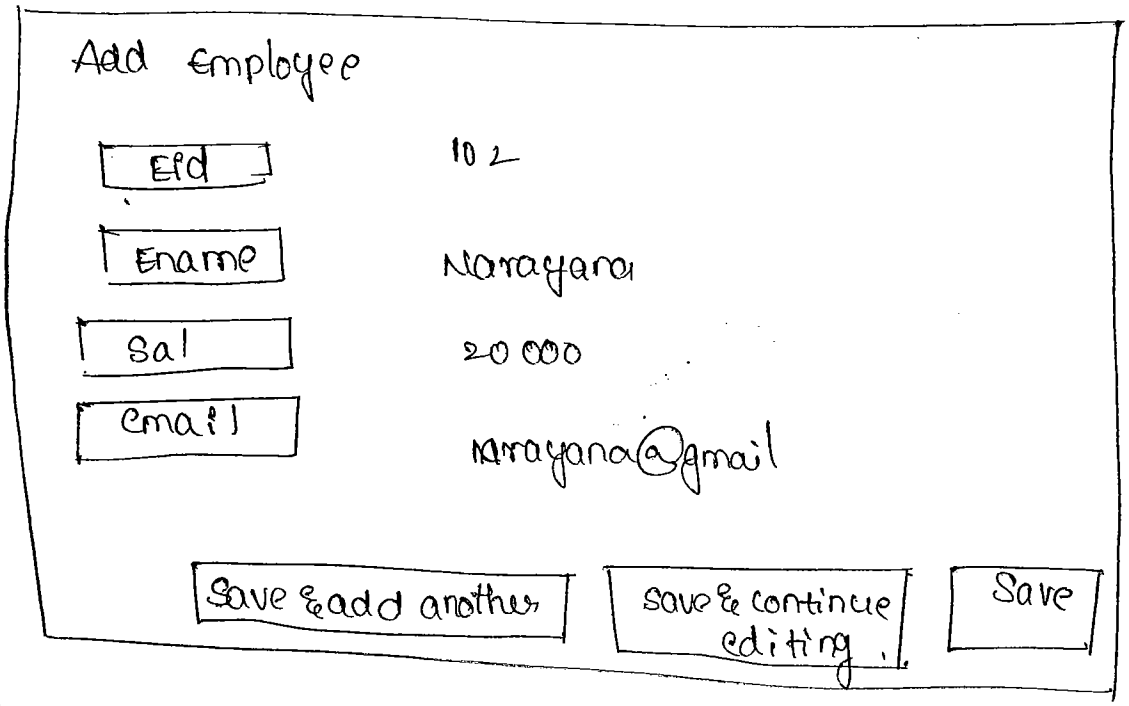
16/06/18

Populating data to the table:-

Step1:- Click on add form of employees model to insert the data.



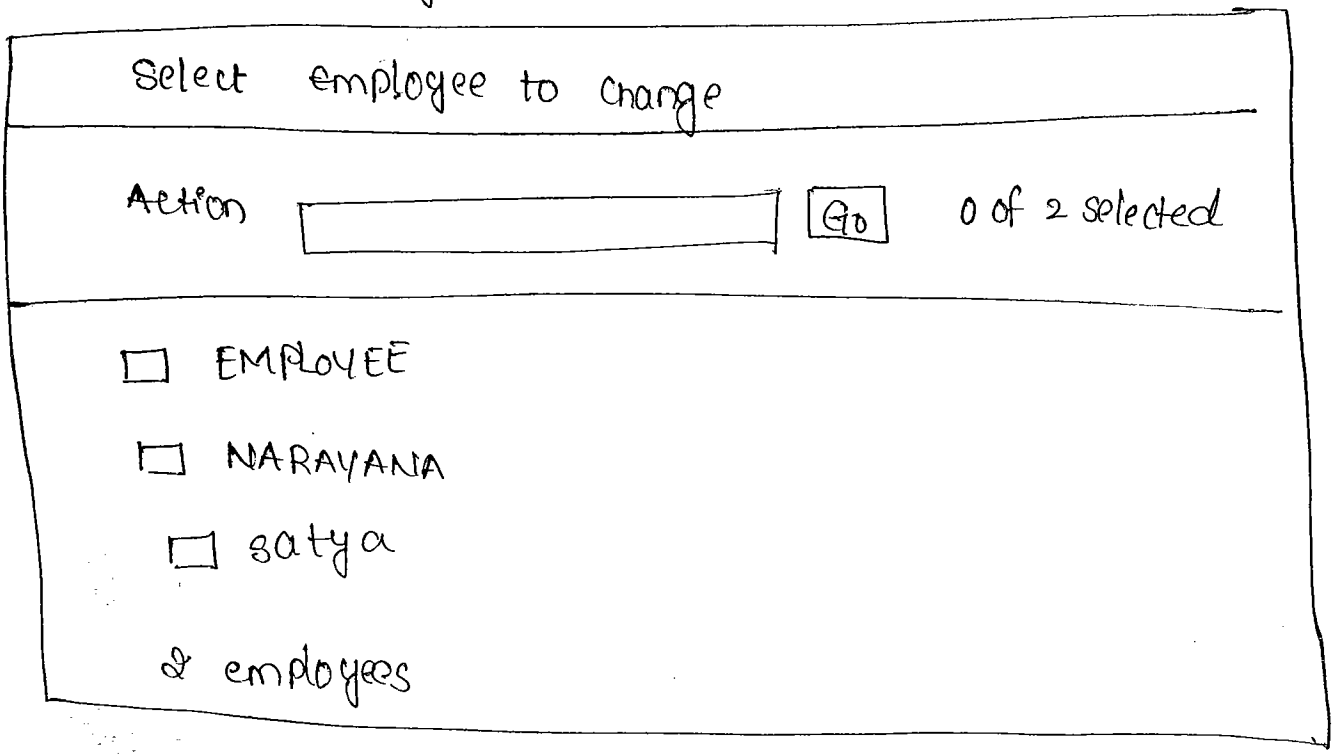
Step2:- click on Save and add another button.



→ click on Save

Step3:-

click on change list of employees model to display all Employees details.



→ Here it displaying only Employees name even through the model has multiple we nud to do some changes at admin.py file to

get all fields in the admin site.

step4 :-

→ go to admin.py file and write the below code.

```
from django.contrib import admin
from .models import Employee.
class AdminEmployee (admin.ModelAdmin):
    list_display = ['eid',
                    'ename',
                    'sal',
                    'email', ]
admin.site.register (Employee, AdminEmployee)
```

Step4:-

→ go to Admin site refresh the Browser

Select Employee to change

Action [ ] Go [0 of 2 selected]

	EID	ENAME	SAL	EMAIL
☐	102	Narayana	20000	narayana@
☐	101	satya	1000	satya@

Step5:-

open mysql & Activate 'dbadmin' database

```
mysql > use dbadmin;
```

- database change

→ Run server selected to display all employees data

mysql > Select * from adminapp_employee;

id	eid	ename	sal	email
1	101	narayana satya	10000	satya@gmail.com
2	102	Narayana	20000	narayana@gmail.com

Diff Blw Delete & Truncate the data:-

→ If we delete the above command by using delete command `adminapp_employee;` then the new record will continue the default id value.

like:- `mysql > delete from adminapp_employee;`

Query ok, 2 rows affected (0.07 sec)

→ Now we give new record in the admin site and Then run the Select statement.

mysql → mysql > select * from admin_employee;

id	eid	ename	sal	email
3	103	raj	1000	raj@gmail.com

* If we truncate above table by using 'truncate table' Command `adminapp_employee;` then the new record will reset from 1 onwards, it will not continue from 4 onwords in the above example...

```
mysql> truncate table adminapp_employee;
```

id	eid	ename	sal	Email
1	101	Siva	40000	siva@gmail.com

1 row in set (0.00sec).

Types of fields:-

(1) Integer field ( )
(2) charfield ( )
(3) Email field ( )
(4) float field ( )
(5) SmallIntegerfield ( )
(6) URLfield ( )
(7) Text field ( )
(8) file field ( )
(9) file path field ( )
(10) one-to-one field ( )
(11) Many to Many field ( )
(12) foreignkey ( )
(13) Datefield ( )
(14) Time field ( )

12/06/18

N → N/w db
O → object db
h → hierachy db
R → Relational db (1989)

→ oracle 12/12 Rules
→ sql 10/12
→ mysql 8.5/12

EO codd

Relationships:-

Django models also (called) Relational management system like all other databases ~~over~~ our django models also support 3 relationships. (follows)

(1) one to one
(2) many to one
(3) many to many.

(1) one to one Relationship:- we use one to one field to implement One to one relationship b/w the models.

→ one to one relationship means The Parent table record must have only one corresponding record in The child table.

→ If we try to take multiple records in child table for a same parent table. record the one To one field with return Error.

Step1:- Open pycharm & clicks on file tab and choose New project.

Step2:- Specify foldername and project name at location option.

Location : E:\Relationalfolder\Relationalpro

Step3:- Expand more settings option and give the application name.

Application name : Relationalapp

→ click on create option.

Step4:-

Open mysql and create a database with name Relationaldb. <mysql> create database Relationaldb;>

Step5:-

go to Settings.py file and configure the "database".

```
DATABASE = {
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
```

```
'NAME' : 'Relational db',
'USER' : 'root',
'PASSWORD' : 'root',
}
}
```

Step 6:-

Go to models.py file.

```
from django.db import models
class Student (models.Model):
    sname = models.charfield (max_length = 20)
    loc   = models.charfield (max_length = 20)
    age   = models.IntegerField ( )
    email = models.emailfield (max_length = 20)
    def __str__ (self):
        return self.name

class Course (models.model):        on delete = (CASCADE)
                                              ↓ for set
    student = models.OneToOneField (Student)   : real
    cname   = models.CharField (max_length = 20)   : NULL=True
    fee     = models.IntegerField ( )
    def __str__ (self):
        return self.name.
```

Step 7:-

Go to Admin.py file (To locate all data in Admin)

```
from django.contrib import admin
from .models import student, course.
```

```
Class Adminstudent (admin.ModelAdmin):
    list_display = ['sname', 'age', 'email', 'loca',
                    'fee']
```

Space

```
class Admincourse(admin.ModelAdmin):
    list_display = ['cname', 'fee']
```

Space

```
admin.site.register(Student, Adminstudent)
admin.site.register(Course, Admincourse)
```

} to register data in admin.py file

Step 8:- Create username and password.

```
E:\Relationalfolder\relationalpro>python manage.py createsuperuser
Username (leave blank to use 'narayana'):
Email address: narayana@gmail.com
Password: 1234abcd
Password (again): 1234abcd.
Superuser created successfully.
```

Step 9:-

Run: makemigration Command.

```
E:\Relationalfolder\Relationalpro>Python manage.py makemigrations
```

Step 10:- Run migrate command to create tables in database

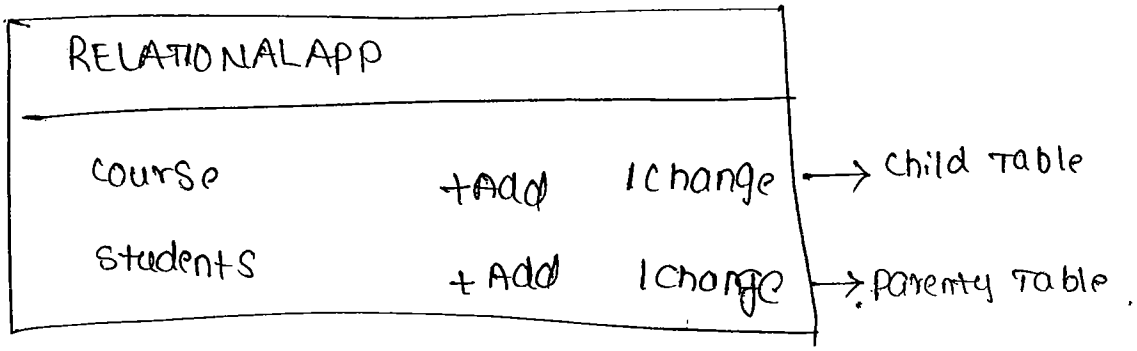
```
E:\Relationalfolder\Relationalpro>python manage.py migrate
```

Step11:- Run the server

```
> python manage.py runserver
```

Step12:-

copy the Ip address and port in any Brower with /admin and give username & pw to login to adminhome page.



Step13:-

Click on add form of student to enter student details.

Add Student

Sname :

loc :

Age :

Email :

Save & add another | Save & continue editing | Save

→ Enter all the details in the same way.

Ex:-

Record-1

Sname : Sai
loc : 21
Age : Hyd
Email : sai@

Record-2

Sname : venkat
loc : 24
Age : Ban
Email : venkat@

Record-3

sname : Saty
loc : Che
Age : 23
Email : saty@

Record-4

Sname : Amit
Age : 24
loc : Pune
Email : Amit@

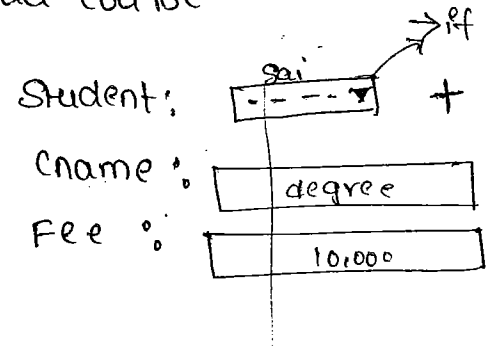
now the table look like

Student

id	sname	age	loc	email
1	Sai	21	Hyd	
2	venkat	24	Ban	
3	Satya	23	Che	
4	Amil	25	Pune	

→ click on add form of course table.

Add course



→ if u click here it display all displays
→ select sai student.

id	cname	fee	student-id
1	degree	10000	1
2	B.Tech	3000	2

→ Add onemore course &

Student: Venkat
Cname: B.Tech
Fee : 30,000 , now the table look like

* Here Student model is called parent model & course model is called child model
* The parent model (student model) has 4 records and course model (child model) has 2 records
* Parent 1 and 2 are called parent records which has child records in the child model.
* So generally we can't delete (or) update the parent records (i.e 1 & 2) in the parent model

because student_id values (1 and 2) are dependent on parent values 1 and 2.

* 3 and 4 are also called parent records but these records have no child records in the child table, so 3 and 4 can be deleted and updated because these records have no dependent records in the child table....

* Generally in other databases we cant update or delete parent records initially bcoz they have 'NO_ACTION' cascading rule as default rule.

* But in Django models the default cascading rule is CASCADE.

Different types of cascading rules in django models:-

(1) DO-NOTHING:- if we set "DO_Nothing" as a default cascading rule for 'on_delete' in the child Model/table then we can delete the parent records (1 and 2).

(2) SET_NULL: If we "set_NULL" = True {syntax} as default cascading rule for 'on_delete' in the child model then it will set NULL value for the child record when we delete any corresponding parent record in the parent table.

(3) SET(value):- If we set SET (value) as default cascading rule for 'on_delete' in the child model then it will set the specified value to the child record when we delete any corresponding parent record.

(4) CASCADE: It means if delete parent record then corresponding child record also delete.

→ It is the default cascading rule in the django models.

18/06/18

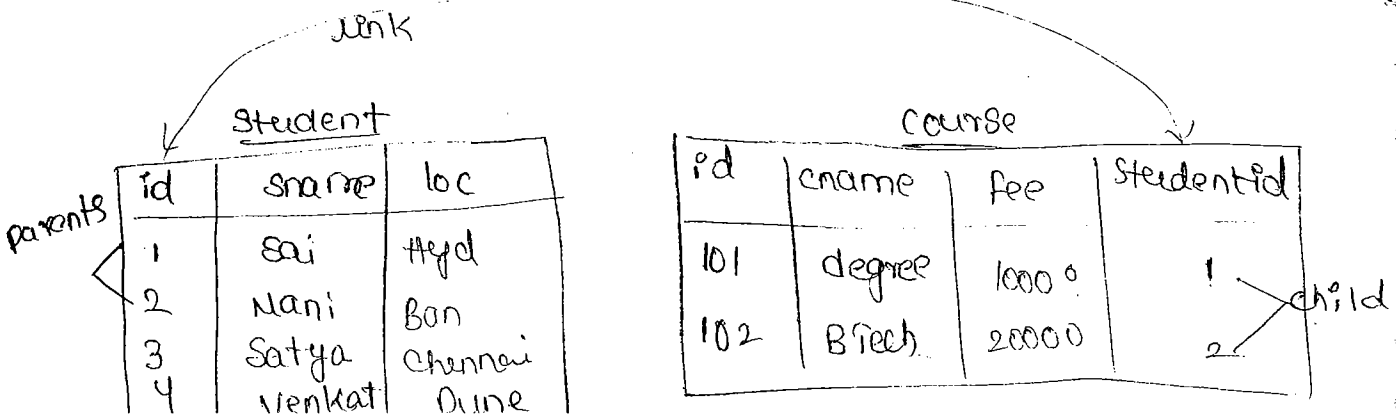
DO Nothing:-

```
class Student(models.Model):
    sname = models.charfield(max_length=20)
    loc = models.Charfield(max_length=20)

class Course(models):

    def __str__(self):
        return self.sname

class course(models.Model):
    Student = models.onetooneField(student, on_dele
                                   te = DO_NOTHING);
    cname = models.charfield(max_length=20)
    fee = models.IntegerField()
    def __str__(self):
        return self.cname
```



* The above example we can't delete parents 1 & 2 in the student table bcoz the parents ~~the~~ 1 & 2 are having the children 1 & 2 in the course table.

* In Django we use DO_NOTHING cascading Rule to avoid deleting parents which are having (corresponding) children.

* In the databases like Oracle, SQL Server we use NOACTION cascading Rule.

* We can delete 3 & 4 parents BCOZ the parents 3 & 4 are having no childrens in the course table.

(2) SET_NULL :-

```
class student (models.Model):
    sname = models.charfield (max_length=20)
    loc   = models.charfield (max_length=20)
    def__str__(self):
        return self.sname

class course (models.model):
    student = models.oneToOneField (student,
                                    on_delete=SET_NULL,
                                    null=True)
    cname = models.charfield (max_length=20)
    fee = models.IntegerField ( )
    def__str__(self):
        return self.cname
```

Student

id	sname	loc
1	Sai	Hyd
2	Nani	Ban
3	Satya	che
4	venkat	Pune

Course

id	Cname	fee	Studentname
101	degree	10000	1
102	B.Tech	20000	2

* Here we can delete parents 1 & 2 also in the student table Bcoz we set on_delete = SET_NULL, If we delete parent 1 in the student table then the corresponding child 1 will be NULL value.

student table

id	sname	loc
3	satya	che
4	venkat	pune

course table

id	cname	fee	S.N
101	degree	10000	NULL
102	B.Tech	20000	NULL

(3) SET (VALUE) :-

```
class student (models.Model):
    sname = models.charfield (max_length = 20)
    loc   = models. charfield (max_length=20)
    def __str__(self):
        return self.sname
class course ( models. model)

    student = models.OneToOneField (student, on_delete=SET(4)
    cname = models · Charfield (max_length =20)
                                        null=true)
    fee = models. IntegerField ( )
    def __str__(self):
        return self.name
```

Student table

id	sname	loca
1	sai	Hyd
2	nani	Ban
3	satya	che
4	venkat	pune

Course table

id	cname	fee	S.Id
101	degree	10000	1
102	B.Tech	20000	2

* Here we can delete the parents 1 & 2 Bcoz we set on_delete = SET(4). if we delete parent-1 in the student table then the corresponding child 01 becomes 4 in the course table.

Student table

id	sname	loc
2	nani	Ban
3	satya	che
4	venkat	pune

course table

id	cname	fee	S.Id
101	degree	10000	4
102	B.Tech	20000	2

* now in student table 2 & 4 parents are having children in the course table.

(4) CASCADE:-

```
class Student (models.Model):
    cname = models.CharField (max_length=20)
    loc   = models.CharField (max_length=20)

    def __str__ (self):
        return self.sname

class course (models.Model):
    Student = models.OneToOneField (student, CASCAD
                                    on_delete = SET(4),
                                    null = True)
```

```
cname = models.charField (max_length=20)
Fee   = models.IntegerField ( )

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

Student table

id	sname	loc
1	sai	hyd
2	Nani	Ban
3	satya	che
4	venkat	pune

course Table

id	cname	fee	S.Id
101	degree	10000	1
102	B.Tech	20000	2

* Here we can delete parents 1 & 2, if we delete either parent 1 or parent 2 then (automatically) the corresponding child record of the course table will be deleted.
* CASECADE means Apply same operation on the child table.
* It is the "default" rule in the django.

Student Table

id	sname	loc
2	Nani	Ban
3	satya	che
4	venkat	pune

course table.

Id	cname	fee	S-Id
102	B.Tech	2000	2

* same operation (student table & course table)

2. ManyToOne Relationship:-

→ Here the child table having multiple records for same parent record in the parent table.

→ we use ~~ManyToOneField~~ ForeignKey → ForeignKey to implement many-to-one relationship.

Step-1:- create folder name 'RelationalFolder' and Project name 'RelationalPro' and Application with 'Relationalapp' in Pycharm.

Step-2:- open Pycharm

New Project ↓ Django → Location E:\RelationalFolder1\RelationalPro

More settings

Template language —

Template folder —

Application name RelationalApp

→ click on create-Button. create

19106118

Step-3:- GoTo my-SQL and create Database.

```
mysql> create database mtodb;
Query OK, 1 row affected (0.00 sec)
mysql> use mtodb;
Database changed.
```

Step4:- go to settings.py file and configure the database

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

NOTE:- If we create a project through pycharm then Django automatically specifies our Application name under INSTALLED_APPS in setting.py file. If we specify manually Then it will through Error.

Step5:- go to models.py file.

```
from django.db import models

class Student (models.model):
    sname = models.charfield (max_length = 20)
    loc   = models.charfield (max_length = 20)
    def __str__ (self):
        return self.sname

Class Course (models.model):
    ~~student = models.charfield (max~~
    student = models.Foreignkey (student)
    cname = models.charfield (max_length = 20)
```

```
    fee = models.IntegerField()
    def __str__(self):
        return self.cname
```

Step6:- go to admin.py file to import the models & register the models with admin site.

```
from django.contrib import admin
from .models import student, couse

class Adminstudent (admin.ModelAdmin):
    list_display = ['sname', 'loc']

class Admincourse (admin.ModelAdmin):
    list_display = ['cname', 'fee']

admin.site.register (student, Adminstudent)
admin.site.register (course, Admincourse)
```

Step7:- go to command prompt run createsuperuser command, and → makemigrations → migrate command → run server.

createsuperuser :-

E:\Relationalfolder1\RelationalPro>python manage.py createsuperuser :

- User name : Narayana
  p/w : 1234 abcd
  conform p/w : 1234 abcd.

makemigrations :-

```
E:\Relationalfolder1\RelationalPro>python manage.py makemigrations
```

migrations for 'Relational-App':

0001_initial.py:

- create model course
- create model Student
- Add field Student to course

migrate :-

E:\Relationalfolder\RelationalPro> Python manage.py migrate

Run Server

E:\Relationalfolder\RelationalPro > Python manage.py runserver.

Steps:- login to the IP Admin site

RELATIONAL APP		
courses	+Add	/ change
students	+Add	/ chang

Step9:- clicks on Add form of the student, to give data.

Add student

(1)

Sname	Nani
loc	Hyd

(2)

Sname	venkat
loc	Bang

(3)

Sname	satya
loc	chennai

save

Table :-

id	Sname	loc
1	nani	Hyd
2	venkat	Bang
3	sat	chennai

Step10:-

click on Addformof course table

Add course

Student:

Cname:

Fee :

Add onemorecourse

id	cname	Fee	Student-id	name
1	Python	1500	1	(nani)
2	Django	200	1	(nani)
3	Java	8000	2	(venkat)
4	oracle	100	2	(venkat)
5	SQL	1500	3	(satya)

* Initially, the parents 1, 2 & 3 in student table can be deleted - because the default cascading rule is CASCADE. To avoid deleting parent records (1,2,3 only) Then we have to change the cascade Rule to DO_NOTHING.

Steps After changing The cascade rules,

Rule (save)

↓

makemigration (cmd)

↓

migrate (cmd)

↓

RunServer (cmd)

↓

admin site (refresh)

↓

Select student to change

↓

Action

↓

Yes, I'm sure (Execute Remotely)

↓

Gotomy sql (select * from relationalpro_student)

↓

select * from Relationalpro_course

ManyToMany Relationship:-

→ In this Relationship multiple child records will depend on multiple parent records.

→ We use ManyToManyField to implement many To many relationship.

Step1:- folder Name- Relational folder

Step2:- Project Name - RelationalPro

Step3:- AppName - RelationalApp

Spt4:- open Relational-folder in Pycharm

Step5:- go to mysql and create 'mtmdb' database.

Step6 :- go to settings.py file and configure the database

Stept:- go to models.py file

```
from django.db import models

Class Author(models.Model):
    aname = models.CharField(max_length=20)
    loc = models.CharField(max_length=20)
    def __str__(self):
        return self.aname.

Class Book(models.Model):
    bname = models.CharField(max_length=20)
    author = models.ManyToManyField(Author)
    bname = models.CharField(max_length=20)
    bCost = models.IntegerField()
    def __str__(self):
        return self.bname.
```

Step 8:- click on 'tools' tab and click on Run manage.py task option. then it opens a new window at the bottom of the Pycharm.

```
manage.py@RelationalPro>
```

now, here we can run any command by giving command name (here we dont use 'python manage.py')

Step 9:-
Run 'makemigrations' command.

```
manage.py@RelationalPro > makemigrations
```

Step 10 :- Run migrate command

```
manage.py@RelationalPro > migrate
```

Step 11 :- Create a superuser.

```
manage.py@RelationalPro>createsuperuser
```

Step 12 :- Run server

```
manage.py@RelationalPro> runserver
```

* now, we just click on IP address link so that it will open a new Browser with IP address.
* we add `/admin/` and enter username & password to login to the admin home page.

Step 13:- goto admin.py file to write the below code:-

```
From django.contrib import admin
from. models import Author, Book.

Class AdminAuthor (admin, ModelAdmin):
    list_display = ['aname', 'loc']

class AdminBook (admin. ModelAdmin):
    list_display = ['bname', 'book']

admin.site.register (Author, AdminAuthor)
admin.site.register (Book, AdminBook)
```

Step14:

now Refresh Adminsite to get the tables.

Relationalapp		
Authors	+Add	/ change
Books	+ Add	/ change

* now It give the data to the Authormodel below.

"Author".

id	aname	loc
1	Amit	Hyd
2	kesav	Bang
3	sunil	che

Step15:- click on Add-form of Books table.

Add book

Author [Amit kesav sunil] + → we can select multiple Authors for same book.

Bname

Bcost

Book Table

id	bname	bcost	
1	Python	1000	→ written By Amit, Sunil
2	Dfango	2000	→ " Kesav, Sunil
3	oracle	500	→ written By Sunil only

Step16:- Dfango will create a new "Intermediate table" B/w Author & Book table.

Book_Author Table

id	bid	aid
1	1	1
2	1	3
3	2	2
4	2	3
5	3	3

Step17:- If I delete Amit record (Id-1) Then In Book_Author Table id-1 will remove (1,1). But in Books Table The Books will not remove.

* If I Sunil record (Id-3) Then In Book_Author Table id's 2, 4 & 5 will remove ([1,3], [2,3], [3,3]).

MetaModel :-

* Meta Model contains Everything Except Columns (or) Fields of Model.
* Meta model is used to manage | controll the main model. like, Ordering the main model based on specified Field changinging the database table name.

Step1 :- folder Name : Metafolder

Step2 :- Project Name : Meta project

Step3 :- APP Name : Metaapp

Step4 :- open Metafolder in Pycharm

Step5 :- go to Models.py

```
from django.db import models

class Emp(models.Model):
    ename = models.CharField(max_length=20)
    loc = models.CharField(max_length=20)
    sal = models.IntegerField()
    def __str__(self)
        return self.ename

class Meta:
    ordering
    class Meta():
        ordering = ['ename']
```

Step6:- Go to Admin.py file and write below code

```
from django.contrib import admin
from .models import Emp

class AdminEmp(admin.ModelAdmin):
    list_display=['ename','loc','sa']

admin.site.register(Emp,AdminEmp)
```

Step7:- go to tools tab and click on 'run manage.py task'

Step8:- run migrate Command

Step9:- run createsuperuser command

```
username : Narayana
password : 123uabcd
```

Step10:- run makemigrations

Step11:- run migrate command

Step12:- run runserver Command and click on IPAddress link

Step13:- login to the Adminsite and enter data to the table

Step14:- The data displays in Accending order.

Select Emp to change

Ename	loc	Sal
Amit	Pune	100000
Narayana	Hyd	10000
satya	Bang	20000

Step 15:- If we make metamodel like below then it displays Emp table in decending order based on same ename field.

```
class Meta():
    Ordering = ['-ename']
```

Step 16:- Now refresh admin site then the table displays like below.

Select Emp to change

Ename	loc	Sal
Satya	Bang	20000
Narayana	Hyd	10000
Amit	Pune	100000

Model Inheritance:-

→ Model Inheritance is like Python Inheritance Concept

→ Django supports different types of model Inheritance.

(1) Abstract Model Inheritance
(2) Multitable Model Inheritance.
(3) Multiple Model Inheritance
(4) proxy Model Inheritance

(1) Abstract Model Inheritance:-

→ In this model Inheritance, the Abstract model will have the fields which are common to other models.

→ Django will not create data base table for Abstract model bcoz it doesn't have any specific data in this model.

Step1:- folder Name : E:\Inheritance folder\Adminmodelpro

Step2:- Project Name : Inheritanceproject.

Step3:- App Name : Abstractapp

Step4:- go to models.py

```
from django.db import models

class common(models.model):
    name = models.charfield(max_length=20)
    loc = models.charfield(max_length=20)

class Employee(models.model):
    sal = models.IntegerField()
    mobile = models.Integerfield()
    def __str__(self):
        return self.name

class customer(models.model):
    age = models.charfield(max_length=20)
    age = models.IntegerField()
    email = models.Integerfield(max_length=30)
    def __str__(self):
```

Step 5:- go to Admin.py file.

```
from django contrib import admin
from . models import Emp

class Admin_Emp (admin.ModelAdmin):
    list_display = ['ename', 'loc', 'sal', mobile]

class Admincustomer (admin.ModelAdmin)

admin.site.register (Emp, AdminEmp)
admin.site.register (Customer, Admincustomer)
```

(Modelspace)

Step 6:- go to mysql and create database

```
mysql > create database abstractdb;
```

```
mysql> use abstractdb;
```

Step 7:- go to setting.py file and configure the database.

```
DATABASE = {
    'default': {
        'ENGINE' : django.db backends.mysql',
        'NAME'   : 'abstractdb',
        'USER'   : 'root',
        'PASSWORD' : 'root',
        }
    }
```

Step6:- go to tools click on run manage.py Task.

Step7:- Run migrate command

Step8:- create superuser

user name: Narayana

Password: 1234

Step9:- make migrations.

```
E:\Inheritancefolder\AbstractModelPro> python manage.py makemigrations
```

0001_customer_employee.py :-

- create model customer
- create model Employee

Step10:- ~~make~~ run migrate command

Step 11:- run the server

```
E:\Inheritance-folder\AbstractModelPro> Python mange.py runserver
```

Step12:- Open Browser login to admin site then admin home page displays like beloco.

Site Administration

ABSTARCTMODELAPP		
Customers	+Add	/change
Employee	+Add	/change

Step 13:- Click on Add form of employee. finally click save.

Add employee

Name : ~~venkka~~ satya

loc : Bang

~~Sal~~ Sal : 10000

~~mobile~~ mobile : 9010509 0

Step 14:- mysql > Show tables;

↓

mysql > Select * from abstractmodelapp_customer;

id	name	loc	age	email
1	venkat	Hyd	30	venkat@gmail.com

mysql > select * from abstractmodelapp_employee;

id	name	loc	Sal	mobile
1	satya	Bang	10000	9010 5090

Step 15:- If two (or) more models are having some common field names then we can create another model / Abstract mode with all these common field names and we should not take same colomns in other models.

Multitable Model Inheritance:-

→ Multitable Inheritance is like multilevel inheritance in Python.

→ In multitable Inheritance, we specify the parent model name in the child model. so that the child model takes all fields from parent model.

→ while giving data, we directly go to the child model and give the data for both child and parent model, bcoz the child model contains all fields of parent model.

step1: create a new database with name `multitabledb`

step2: go to settings.py file to configure the database.

step3:- go to models.py file

```
from django.db import models
class Grand_father(models.Model):
    gname = models.charfield(max_length=20)
    age = models.IntegerField(max)
    def __str__(self):
        return self.gname
class father(models.Model):  # Grand_father ):
    fname = models.charfield(max_length=20)
    job = models.charfield(max_length=20)
    def __str__(self):
        return self.fname
```

```
Class son (father):
    sname = models.CharField (max_length=20)
    course = models.CharField (max_length=20)
    def __str__(self):
        return self.sname
```

step 4:- go to Admin.py

```
from django.contrib import admin
from .models import Grand_father, father, son

Class admin Grand_father (admin.ModelAdmin):
    list_display = ['gname', 'age']

class Admin father (admin.ModelAdmin):
    list_display = ['fname', 'job']

Class Admin son (admin.ModelAdmin):
    list_display = ['sname', 'course']

admin.site.register (Grand_father, AdminGrand_father)
admin.site.register (father, Admin-father)
admin.site.register (son, Adminson)
```

step 5: run migrate

step 6:- create superuser

step 7:- run make migrations

0008_ father_ grandfather_ son.py :

- create model Grand_father
- create model father
- create model son.

Step8: Run migrate

Step9: Run server

Step10:- Open Browser and login admin site

Step11:- Click on addform of son model

Addson

Gname : satya
age : 90
} These too will insert in Grandfather table.

Fname : venkat
Job : SW
} father table

Sname : sai
course : Python.
} son table.

Step12: Click on save & another button.

Add son :-

Gname : satya raj
age 90
Fname venkat raj
Job Police

Sname sai raj
course django

click on save.

Step 13:- go to mysql data base.

```
mysql > select * from abstractmodelapp_father;
```

grand_father_ptr_id	fname	job
1	venkat	sw
2	venkat raj	police.

```
mysql > select * from abstractmodelapp_Grand_father;
```

id	gname	age
1	satya	90
2	satyaraj	90

```
mysql > select * from abstractmodelapp_son;
```

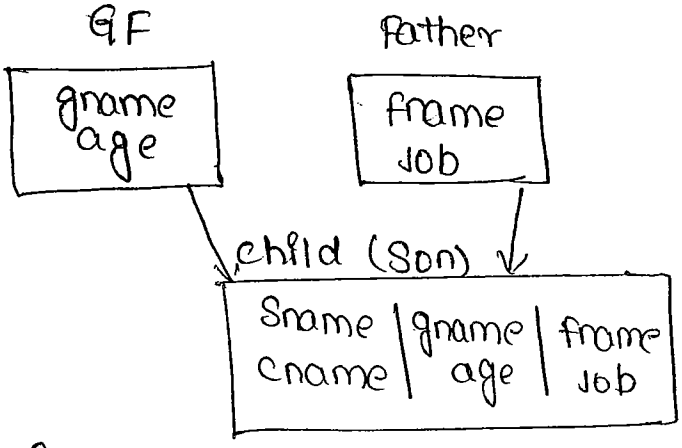
father_ptr_id	sname	course
1	sai	python
2	sai raj	django

23/06/18

Multiple Model Inheritance :-

→ Multiple Model Inheritance will work like multiple inheritance in Python.

→ In this inheritance, we derive the fields from multiple parents into single child model.



Step 1 :- Folder name : multiplefolder.

Step 2 :- Project name : multiplepro

Step 3 :- App name : multipleapp

Step 4 :- Open multipro in pycharm.

Step 5 :- go to models.py file and write the beloco code.

```
from django.db import models
class Grand_father (models.Model):
    gid = models.AutoField (primary_key=True)
    gname = models.CharField(max_length=20)
    age = models.IntegerField ( )
    def __str__ (self):
        return self.gname
```

```
class Father (models.Model):
    fid = models.AutoField(primary_key=True)
    fname = models.CharField(max_length=20)
    job = models.charfield(max_length=20)
    def __str__(self):
        return self.fname

class son (models.Model):
class son (Grand_father, Father):
    sname = models.charfield(max_length=20)
    cname = models.charfield(max_length=20)
    def __str__(self):
        return self.sname
```

Step 6:- go to Admin.py file and write the below code.

```
from django.contrib import admin
from .models import Grand_father, Father, son

class Admin Grand_father (admin.ModelAdmin):
    List_display = ['gname', 'age']
class Adminfather (admin.ModelAdmin):
    list_display = ['fname', 'job']
class Admin son (admin.ModelAdmin):
    list_display = ['sname', 'cname']
```

admin.site.register(Grand_father, AdminGrand_father)
admin.site.register(father, Adminfather)
admin.site.register(son, Adminson)

Step:7 Open mySQL create data base with name multipledb

```
mysql> create database multipledb;
```

Query ok, 1 row affected (0.02 sec)

```
mysql> use multipledb;
```

database changed.

Step8:- go to setting.py and configure the database

```
DATABASE = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'multipledb',
        'USER': 'root',
        'PASSWORD': 'root',
```

Step9:- go to tools Tab click 'Run manage.py Task' (ctrl+Alt+R)

Step10:- run migrate command.

Step11:- create superuser
(no space)

username : Narayana
mail : Narayana@gmail.com
p/w : N3u0bcd

Step 12:- Run makemigration (no space) command.

0001_initial.py :

- create model father
- create model Grand_father
- create model Son.

Step13:-Run migrate command

Step14:-Run server, Then click on Ip adaress link so that it opens the Browser with current address.

Step15: Enter username & Password to login into Admin home page.

MultipleApp Administartion

MultipleApp

father	+Add	/change
Grand-Father	+Add	/change
Son	+Add	/change.

Step16:- click on addform of son model to give the data for '3' models (father, Grand_father, Son)

Add Son

Gname	Satya
age	90
fname	Durga
Job	SW
Sname	Sai
Cname	Python

Save & addanother

Add son

Gname → satya raj
age → 91

Fname → Durgaraj
Job → Police

Sname → Sairaj
Cname → Django.

Save

Now we can see the data of 3 models in Adminsite & database.

Step 17:- go to mysql database

mysql > show tables;

mysql > select * from multipleapp_grand_father;

gid	gname	age
1	satya	90
2	satyaraj	91

mysql > select * from multipleapp_father;

Fid	Fname	Job
1	Durga	SW
2	Durgaraj	Police

my sql > Select * from multipleapp_son;

grand_father_ptr_id	father_ptr_id	Sname	Cname
1	1	Sai	Python
2	2	Sairaj	Django

25/06/18

Proxy Model Inheritance:

→ Django supports proxy model Inheritance to control the main model in their models.py file.

→ proxy means controlling

-> we can control the main model (non-abstract) by inserting, updating and deleting the data through proxy model.

→ we set proxy = True to make a model as proxy model.

Step 1: foldername : Proxy

Step 2: Project name : Proxypro

Step 3: App name : Proxyapp

Step 4: open proxy folder in the Pycharm.

Step 5: go to models.py file and write the below code.

```
from django.db import models

class Emp(models.Model):
    eno = models.IntegerField()
    ename = models.CharField(max_length=20)
    sal = models.IntegerField()
    loc = models.CharField(max_length=20)

class Empproxy(Emp):
    class Meta:
        proxy = True
```

step6 :- go to Admin.py file and write the below code

```
from django.contrib import admin
from .model import Empproxy, Emp

class Adminemp(admin.ModelAdmin):
    list_display = ['eno', 'ename', 'sal', 'loc']

class Adminempproxy(admin.ModelAdmin]
    list_display = ['eno', 'ename', 'sal', 'loc']

admin.site.register(Emp, Adminemp)
admin.site.register(Empproxy, adminempproxy)
```

step7 :- go to tools tab & click on Run manage.py (Ctr+Alt+R)

step8 :- Run migrate command

```
Python manage.py migrate
```

step9 :- Run create superuser commagp

```
Python manage.py createsuperuser
```

User name : Narayana

Password : 1234abcd

step10 :- Run makemigrations command.

```
Python manage.py makemigrations
```

Step11:- Run migragete command

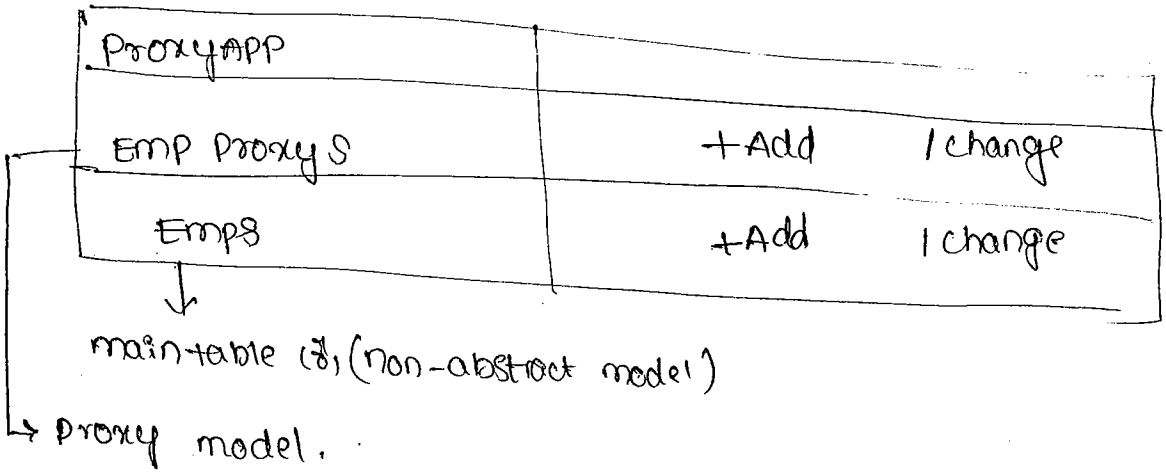
Step12:- Run Servercommand

```
Python manage.py runserver
```

Step13:- Take IP address and go to Browser and paste IP address along with /admin.

Step14:- Enter username & plw to login to Adminhome page
(Narayana) (123uabcd)

Proxyapp Administration



Step15:- now we use proxy model to insert / update / delete the data in the main-table

Step16:- click on Addform of proxy model.

Add emp Proxy

```
eno   : 101
ename : sai
sal   : 10000
loc   : hyd
```

Save & add another

Eno : 102

Ename : Satya

Sal : 20000

loc : Bang

Save

Step17:- click on change list of proxy model to display all the data.

Eno	Ename	SAL	loc
102	Satya	20000	Bang
101	Sai	10000	Hyd

Step18:- If we click on change list of main model then he display same data.

Eno	Ename	SAL	loc
102	satya	20000	Bang
101	sai	10000	Hyd.

Step19:- go to proxy model and update the SAL of Satya.

* click on Eno - 102 ☑ to open all the details of satya (102) so we can update the values.

Eno : 102

Ename : Satya

Sal : 25000

loc : Hyd

Save

* now we can see the same salary of satya in both main and proxy models.

Step 20:- go to proxy model and select 102 and "Delete selected emp proxys" in Action and click on go.

* Then click on Yes. I'm Sure.

Step 21:- now 102 is deleted in proxy model and also in main model.

Using ALL Relationships in a Project:-

Step1:- folder name :- Relationfolder

Step2:- Project name :- Relationpro

Step3:- App name :- Relationapp

Step4:- go to mysql database and create data base with name Relationdb.

```
mysql> create database relationdb;
```

Query OK, 1 row affected (0.02 sec)

```
mysql > use relationdb
```

database changed.

Step5:- go to settings.py file and configure the database.

```
DATABASE = {
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
        'NAME' : 'relationdb'
```

'USER NAME' : 'root'

'PASSWORD' : 'root'

Step6:- Go to models.py file and write below code.

```
from django.db import models

class Publisher (models.Model):
    Pname = models.CharField (max_length=20)
    ploc = models.CharField (max_length=20)
    email = models.EmailField (max_length=20)
    def __str__(self):
        return self.Pname
Class Author (models.Model):
    aname = models.CharField (max_length=20)
    age = models.IntegerField ( )
    income = models.IntegerField ( )
    def __str__(self):
        return self.aname
class Student (models.Model):
    sname = models.CharField (max_length=20)
    location = models.CharField (max_length=20)
    fee = models.IntegerField ( )
    def __str__(self):
        return self.sname
class Book (models.Model):
    Publisher = models.ForgeinKey (Publisher)
    author = models.manyToMany (Author)
    student = models.oneTOMany (Student)
    bname = models.Charfield (max_length=20)
    bcost = models.IntegerField ( )
    def __str__(self):
```

go to admin.py file and write below code.

```
from django.contrib import admin
from.models import Publisher, Author, Student, Book

class AdminPublisher (admin.ModelAdmin):
    List_display = ['pname', 'ploc', 'email']
class AdminAuthor (admin.ModelAdmin):
    List_display = ['aname', age', 'income']
class AdminStudent (admin.ModelAdmin):
    List_display = ['sname', 'location', fee]
class AdminBook (admin.ModelAdmin):
    List_display = ['bname, bcost']

admin.site.register (Publisher, AdminPublisher)
admin.site.register (Author, AdminAuthor)
admin.site.register (Student, AdminStudent)
admin.site.register (Book, AdminBook)
```

Step7:- run migrate command

Step8:- create super user
→ username : narayana
→ pls : 1234 abcd.

Step9:- run makemigration command

Step10:- Again run migrate command

Step11:- run server

26/06/18

Step12 :-

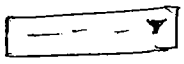
Site administration

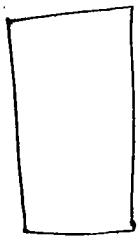
RELATIONALAPP

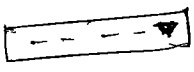
Authors	+Add	change
Books	+Add	change
Publishers	+Add	change
Students	+Add	change.

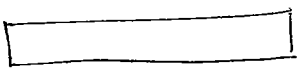
* we can give the data to the all the models (Pub, Author model) Through Books model.
* Click on +Add form of Books model.

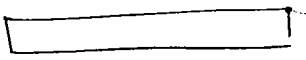
Add Book

Publisher  +

Author  +

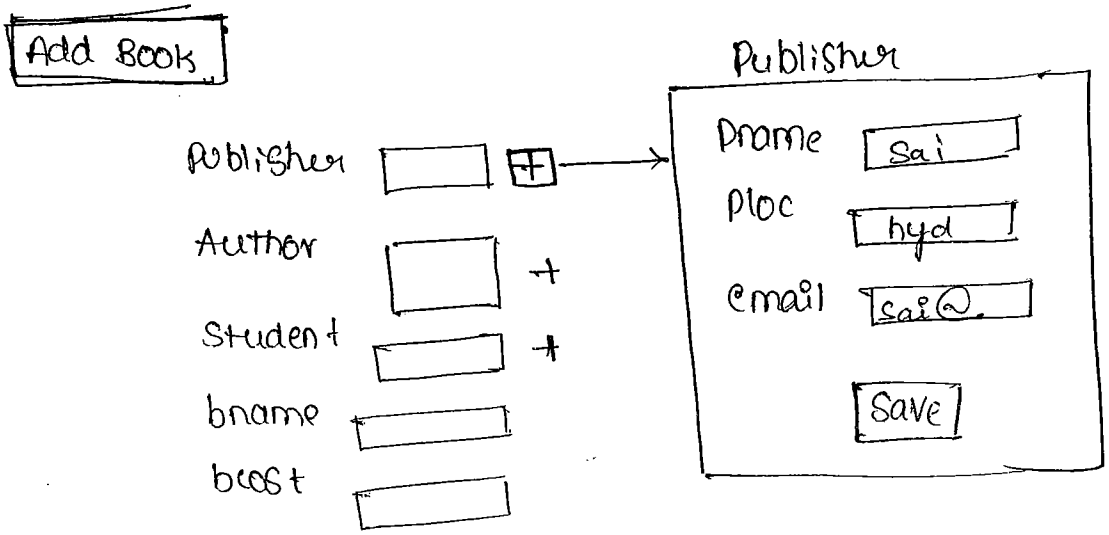
Student  +

Bname 

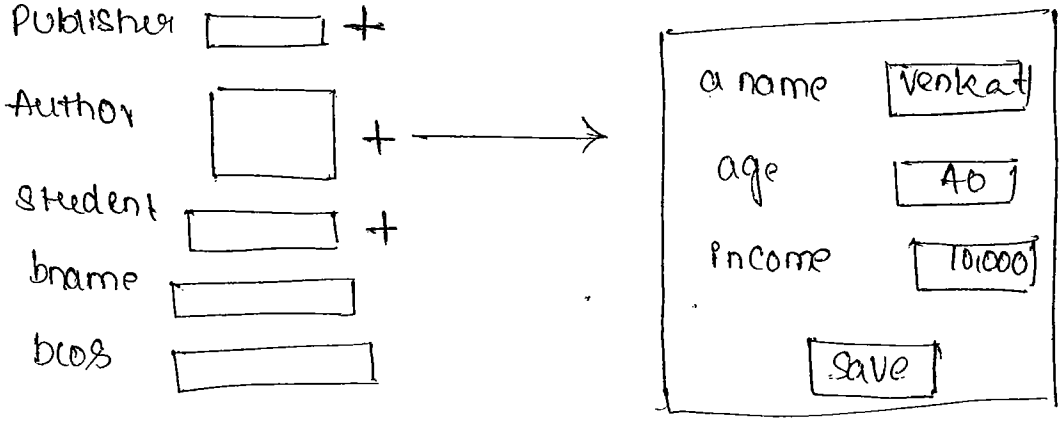
B cost 

Step 13:-

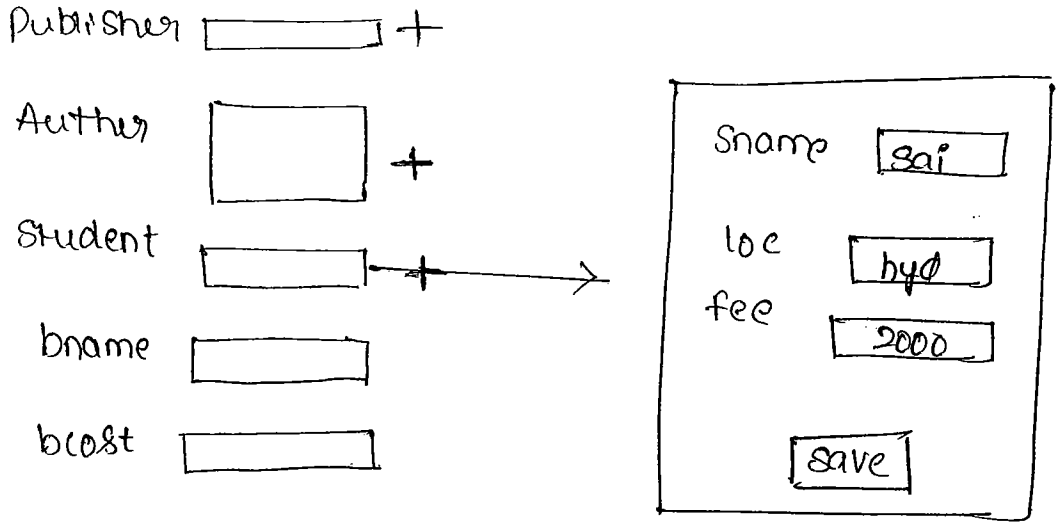
* from this model we can give data to the publisher model, If we click on + symbol the publisher field in the Book model Then it opens publisher model in the new window.



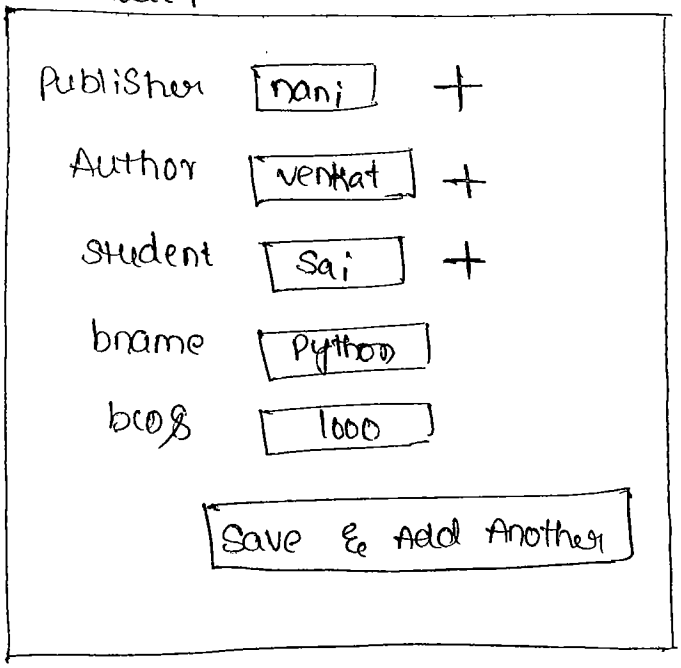
* from this model can also give the data Authors model by clicking + Symbol of Author field in the Author Field



* from the same model can also give the data to Student model by clicking + Symbol of student field in the Book model.



* we also give the data to the bname & b cost fields of Book model.



* we will repeate the above 4 steps to give multiple records to all the models.

Step14:- click on change list of publishers model to see the data

Select Publisher to change

Action

PNAME	PLOC	EMAIL
satya	Bang	satya@gmail
Nani	Hyd	Nani@

Step15:- click on change list of Authors model to see the data.

Authors

ANAME	AGE	INCOME
Hemanth	45	20,0000
venkat	40	10,0000

Step16:- click on Students model to see the data (change list)

SNAME	LOCATION	FEE
Sai Nath	Bang	2000
Sai	Hyd	2000

Step17:- click on Book model to see the data

BNAME	BCOST
django	2500
Python	1000

27/01/18.

Uploading The files & images in the Adminsite :-

* we can upload any number of files and images into the admin site.

eg :-

* all TCS branches will maintain one portal where they will upload employees profile and image, download profile from the portal and they will conduct interview to the employee.

eno	ename	cloc	experiences	status	Resume	profile pic

step1 :- folder Name :- filesfolder

step2 :- project Name :- filespro

step3 :- App Name :- filesapp

step4 :- open files folder in the pycharm.

step5 :- go to fileapp/models.py and write the below code,

```
from django.db import models
Class student(models.Model):
    sno = models.integerfield( )
    sname = models.charfield (max_length=20)
    sloc = models.charfield (max_length=20)
    fmag = models.imagefield (upload_to='images')
    profile = models.profilefield (upload_to='files')
```

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

Step 6:- go to setting.py file.

```
import os
BASE_DIR = os.path.dirname(os.path.dirname
                (os.path.abspath(__file__)))
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Step 7:- go to URLS.py file

```
from django.conf.urls import url
from django.contrib import admin
from django.conf import settings
from django.views.static import serve
urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
if settings.DEBUG:
    urlpatterns += [
        url(r'^media/(?P<path>.*)$', serve,{
            'document_root': settings.MEDIA_ROOT,
        }),
    ]
```

step 8:- go to the location where manage.py file is located and type cmd, to open command prompt with current location.

```
E:\Filesfolder\filespro>
```

step 9: run migrate command

```
> python manage.py migrate
```

step 10:- create superuser command

```
>Python manage.py createsuperuser
```

User name : mounika
P|w : 1234abcd

Step 11 :- Run makemigrations command.

```
> Python manage.py makemigrations
```

Step 12 :- again Run migrate command.

Step 13 :- Run server command

```
>Python manage.py runserver
```

copy the IP address & paste in the Browser with admin site.

Step 14: click on add form of a Student to enter the details.

sno [10]

sname [satya]

sloc [Hyd]

image [choosefile] python.png

Profile [choosefile] Resume_python.doc

Step15:

Click on changelist of student model. [Save]

to see the details of student.

125.0.0.1:8000/admin/Releapp/Student

sno	sname	sloc	Image	files
10	satya	Hyd	images/python.png	files/Resume_python doc

Step16:- "we can download the profile by clicking on the profile name."

* "we can also save the image in local mission by click on the image name."

(1) what is a webframework?

(2) what are the types of webframeworks?

(3) what is django webframework.

(4) what are the advantages of django frameworks.

(5) what are the features of django frameworks.

(6) where did Django frameworks developed.

(7) Does django frameworks follows MVC & MVT architecture.

(8) what is the difference b/w MVC & MVT

(9) what is django project.

(10) what are the different types of files available in the django project?

(11) what is an application?

(12) what are the different files available in the application?

(13) what is the purpose of __init__.py file.

(14) what is the wsgi.py file.

(15) what is view in the Django?

(16) what is difference b/w HTTP request & HTTP response?

(17) what is purpose of urls.py file?

(18) where do we map the urls with corresponding views?

(19) what are the different types of field types of field type in model?

(20) what is the model in django.

(21) what is the field name in model?

(22) what is the base class for all user defined models?

(23) what is string representation in the model?

(24) where do we configure the database in the django project? → setting.py

(25) what are different types of databases which are supporting officially?

(26) what is the default database in django project?

(27) what are different types of relationships in django?

(28) How to implement one to one relationship?

(29) How to implement many to one relationship?

(30) How to implement Many to many relationship?

(31) what are the different CASCADING rules in django model?

(32) what is the default CASCADING rule in Django model.

(33) How to set null value for child record when the corresponding parent record is deleted?

(34) How to avoid deleting parent record when they have corresponding child record?

(35) How to give any default value for child record when we delete corresponding parent record?

(36) what is a manager in the Django mode?

(37) what is the default manager for every model in the django?

(38) what are the different model inheritance in the django.

(39) what is the abstract model inheritance?

(40) what is the purpose of abstract model inheritance?

(41) Does django create a datable table for abstract model or not?

(42) what is the multiple inheritance?

(43) what is multiple inheritance?

(44) what is difference b/w multi-table and multiple inheritance?

(45) what is a proxymodel inheritance?

(46) how to get all model fields in the admin site?

(47) how to create username and password for adminsite?

(48) what is different types of permissions for all users in the admin site?

(49) what is the difference b/w staffstatus (active, login, super) and superuser?

(50) If we want remove any user temporarily in the admin site, then what do we do? (Active)

29/06/18

views.py → HTML
render = ?

Templates :-

HTML coding.

Displaying table data in the Browser :-

Step1: folder name : BrowserData
Step2: project name : Browserpro
Step3: App name : Admin App
Step4 :- open Browser data folder in the Pycharm.
Step5 :- open urls.py file and write the below code.

```
from django.conf.urls import url
from django.contrib import admin
from adminapp.views import index, details

urlpatterns = [
    url(r'^admin/', admin.site.urls)
    url(r'^$', index),
    url(r'^adminapp/info/(\d+)/$', details)
]
```

Step6 : go to views.py file and write the belowcode.

```
from __future__ import unicode_literals
from django.shortcuts import render.
from django.http import HttpResponse
from . models import employee
```

#for displaying only first name and lastname as a list.

```
def index(request):                    <index file for all Employee
    response = HttpRese()
    response.write("<html><body>\n")
    response.write("<h1> Employees details </h1>")
    response.write("<hr>")
    elist = Employee.objects.all()
    for e in elist:
        link = "<a href=\'adminapp\info\%d'> "%(e.id)
        response.write("%s<li>%s   %s </a></li>"
            %(link, e.first_name, e.last_name))
    response.write("<br></body></html>")
    return response
    Break

def details(request, eid="0"):         <detail file for all details of perticular Employe
    response = HttpResponse()
    response.write("<html> <body>\n")
    try:
        e = Employee.objects.get(id=eid)   <- search eid
        response.write("<h1>Details for Employee %s</h1><hr\n"
            %e.first_name)
```

```
response.write("<li>first Name : %s</li><br>" %e.first_name)
response.write("<li>last Name : %s</li><br>" %e.last_name)
response.write("<li> company : %s </li><br> " %e.company)
response.write("<li> location : %s </li><br> " %e.loc)
response.write("<li> salary : %s </li><br> " %e.sal)
except Employee.DoesNotExist: response.write("Employee
                    Not found")
response.write("</body></html>")
return response.
```

Step 7 :- go to models.py file and write below code.

```
from django.db import models.

class Employee(models.Model):
    firstname = models.CharField(max_length=20)
    last_name = models.CharField(max_length=20)
    company = models.CharField(max_length=20, blank=True)
    email = models.EmailField(blank=True)
    sal = models.IntegerField()
    loc = models.CharField(max_length=20)

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

step8:- go to Admin.py file and write below code.

```
from django.contrib import admin
from .models import employee

Class Adminemployee (admin.ModelAdmin):
    list_display = ["first_name", "last_name", "company
                    "loc", "emai", "sal"]
admin.site.register( employee, Adminemployee)
```

step9:- go to data base and create databasename Browser data

```
mysql > create database Browserdata
                         database name
mysql > use Browserdata
    database changed
```

step10:- go to settings.py file and configure the database and specify application name under INSTALLED APPS

```
DATABASE = {
    'Default' : {
    'ENGINE' : 'django.db.backends.mysql',
    'NAME' : 'Browserdata',
    'USER' : 'root',
    'PASSWORD' : 'root',
```

```
INSTALLED_APPS = {
        'django.contrib.admin',
        ---
        ---
        ---
    'adminapp',
}
```

STEP11:- Run make migrations command 30/06/18

STEP10:- Run migrate command

STEP12:- create Superuser

STEP13:- Run Server

STEP14:- Copy the IP address and paste in the Browser along with /admin.

Employee +Add | change

→ clicks on addform to give the data.

Add Employee

first name	Narayana
Last name	Python
company	TCS
Email	pythonnarayana@
Sal	10,000
20C	Hyd

Save & Add another

→ Repeate the above step for multiple employees data.

(2) first name venkat
last name Thota
company wipro
Email venkat@gmail.com
Sal 20,000
loc Bang.

save & Add another

(3) first name Roshan
Last name K
Company ——
Email ——
Sal 30,000
loc mombai.

save & add another

(4) first name Kumar
Last name S
Company TCS
Email: Kumar@gmail.com.
Sal 15,000
loc Hyd

save & addanother

(5) Firstname santhosh
last name K
Company
Email
Sal 30,000
loc mombai

(6) First name Veni

Last name B

Company TCS

Email

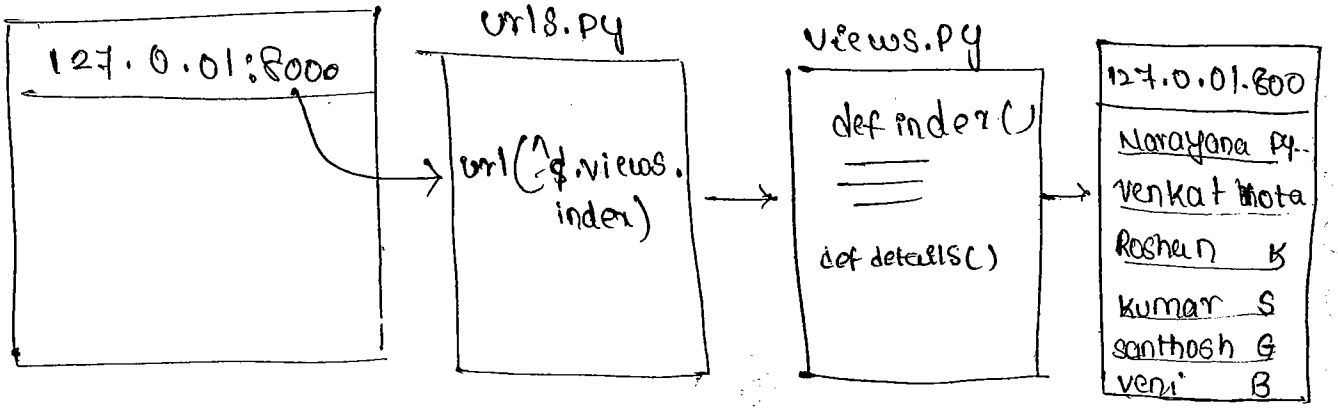
Sal

Loc

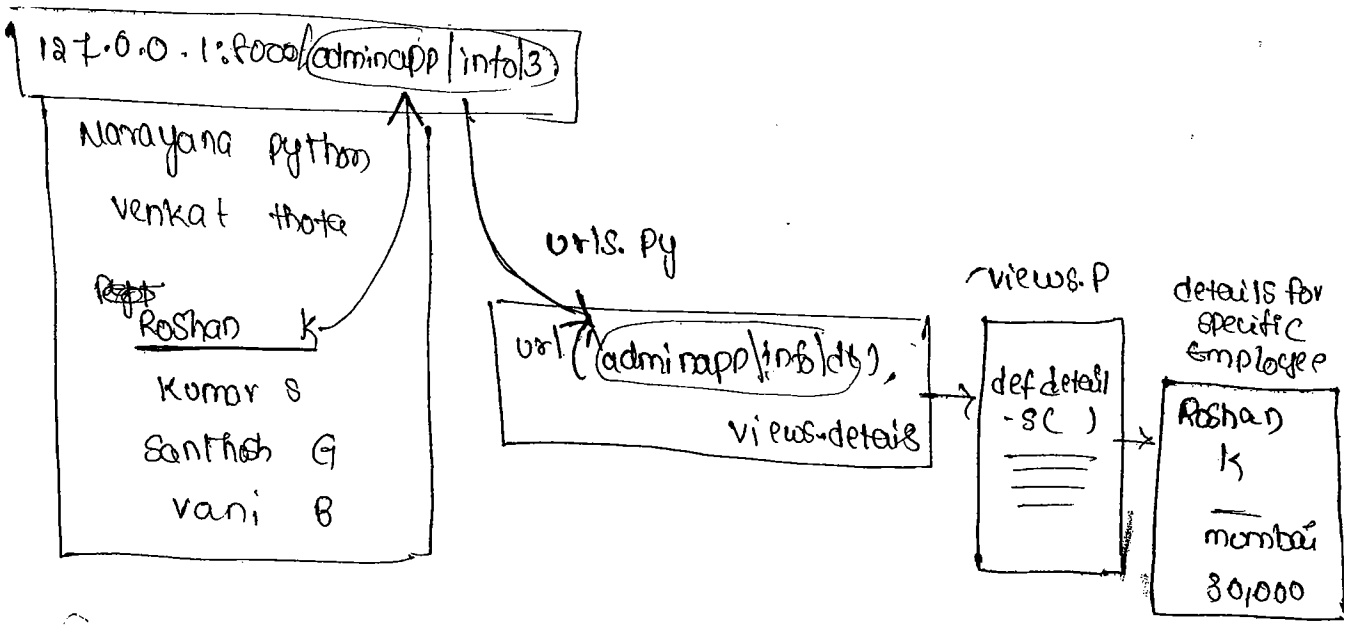
Step15:- open a new browser and specify IP address → click enter.

Employee Details

- Narayana python
- Venkat thota
- Roshan K
- Kumar S
- Santhosh G
- Veni B



* If we click on any specific employee name then it displays all details of that specified employee.



Details for Roshan Employee

First name : Roshan
Last name : k
Company : ____
Location : mumbai
Sal : 30,00

21/7/18

Manager in Model:-

Step 1:- folder name :- managerfolder

Step 2:- project name :- managerpro

Step 3:- app name :- managerapp

Step 4:- go to models.py file

```
from django.db import models

Class Employee (models.Model)
    ename = models.CharField (max_length = 20)
    sal = models.IntegerField ( )
    loc = models.CharField (max_length = 20)
    obj1 = models.manager( )  # changing the manager Object to obj1
    def __str__ (self) :
        self.ename
```

Step 5:- go to Admin.py file

```
from django.contrib import admin

from .models, import Employee

class AdminEmployee (admin.ModelAdmin):
    list_display = ['ename', 'sal', 'loc']
```

admin.site.register (Employee, AdminEmployee)

Step6:- go to database and create database with name manager db.

```
>> create database managerdb;
   row affected
   use managerdb;
```

Step7:- go to setting.py file configure database.

```
DATABASE = {
    'default': {
        'ENGINE': 'django.db.backend.mysql',
        'NAME': 'managerdb'
        'USER': 'root',
        PASSWORD: 'root'
```

Step8:- Run migrate command,

```
>> python manage.py migrate
```

Step9:- create superuser

```
>> python manage.py createsuperuser
   Username:
   Password:
```

Step10:- Run makemigration Command

>> python manage.py makemigration

Step11:- Run migrate Command.

>> python manage.py migrate

Step12:- Run the shell command

>> python manage.py shell

>> from managerapp.model import Employee

>>> a = Employee.object.create ('ename = 'sai', sal = 10000, loc = 'Hyd')

*It will throw an Attribute Error Bcoz we change manager name from object to obj-1 so we should use obj1 as a manager.

>> b = Employee.obj1.create ('ename = 'sai', sal = 10000, loc = 'Hyd')

go to database & check data

mysql > Select * from managerapp_employee ;

Definition of Manager :-

→ A manager is an Interface B/w model & corresponding methodes.

→ A model has one manager by default.

→ The default manager in every model is object.

→ we can also change the manager name of a specific model in that model only by using models.manger( )

→ After changing the managername then we cannot use objects, we should use the new managername.

FieldName Restrictions :-

quit()

① we cannot use python keywords as a field name.

② we cannot use special characters in a field name Excep _ (underscore).

③ we can use multiple underscores seperately but not in a Sequence.

31/07/18

views templates html

Templates :-

→ Templates are mainly used to seperate the html coding from the views.py file.

→ By default, views.py file allows both python coding and also html coding.

→ If we use both Python and html in single view.py file Then it becomes very difficult to manage the file.

→ generally every project has both django developers and also web developers for same project, django developers are responsible to manage python coding and these django developers may not know html coding.

→ Web developers are responsible to manage Presentation part (html coding) and these web developers may not know Python coding, in this process, it very difficult to maintain single file by both django and web developers.

→ Thats why we use Templates to manage html coding seperately from the views.py file.

→ We can write The html coding directly by using variables and tags or we can write the html code in the seperate file and then we take (use) that file.

* If any Text which surrounding by double curlybraces Then that is called variable.

Variable says "insert_something"

eg:- {{a}}

{{name}}

* If any Text surrounding by one curliebrace and one modulus operator Then i.e called Tag.

Tag says "do_something"

Eg :- {% if %}

{% name %}

* We need to fallow The below steps to create a Template (In The Shell).

(1) creating a Template:-

→ A Template is a string which contain one (or) more variables.

→ We Import Template from django.template To create user defined templates.

→ we Take the values from context to insert into The variables in the template.

>>> python manage.py shell

>>> from django.template import template

>>> t1 = Template ("My name is {{ename}} I am working in {{com}} and location is {{loc}} and I am getting {{sal}}")

→ here t1 = Template which contains a string with different variables.

→ now we need context to get the values for these variables.

(2) creating context :-

>>> From django.template import Context

>>> C1 = Context ({"ename": "sal", "com": "TCS", "loc": "hyd", "sal": 10000})

→ Context is a dictonary which contain multiple key value pairs.

→ We take the variables of Templates as keys in the context.

→ We assign required values to the keys in the context.

→ Then in directly the values of this keys to assigned map to variables in Templates, To map the values of keys to the variables in the Template we should use Render function.

→ In this exaple C_1 is a Context which contains different values for all the variables.

(3) Render ( ) :- (go to another file) request

```
>>> x = t1.render(c1)
>>> print(x)
```
↳ argument (context)

My name is Sai, I am working in TCS and location is Hyd and I am getting 10000.

→ render function allows context name as an argument.

→ we use Template name.render(contextname)

That means it will map the values of keys to the context to the variables of Template.

Template system :-

→ we can create html file in the project and then we have to specify the filepath in the Template option in the settings.py file.

<class = capital letters>

↓

go to Template option.

04/07/18

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
        },
    },
```

BACKEND :- It is a dotted python path to specific template engine.

* Django support mainly two types of engine (server name)
  (1) DjangoTemplates
  (2) Jinja2

DIRS : It is a list of directories where our HTML files saved.

→ By default the Django gives a path to the Templates folder.

os.path.join (BASE_DIR, 'templates')

APP_DIRS :- By default it is SET to TRUE that means the Template engine will go to all installed Application and gets HTML files.

→ If we set Falls then the template engine will not go to installed apps.

template project :-

Step 1 :- folder name :- Templatefolder
project name :- pro 2
App name : app 2

Step 2 : Open Template folder in Pycharm

Step 3 :- Specify Appname under INSTALLED_APPS

Step 4 :- go to views.py file and write code.

```
from django.http import HttpResponse
def home_page (request):
    context = {
    'name': 'Dharsana',
    'content': 'Welcome to Home page'
    }
    return render (request, 'hello.html', context)
def contact_page (request):
    context = {
    'name': 'Dharsana',
    'content': 'welcome to contact page'
    }
    return render (request, 'hello.html', context)
def about_page (request):
    context = {
    'name': 'Dharsana',
    'content': 'welcome to about page'
    }
    return render (request, 'hello.html', context)
```

Steps:-

click on main project folder (pro2) and select new → directory → give the name Templates

Step 6:-

Click on Templates folder → new → html file → give any name (ex. hello). → ok.

↓

(Then it given data automatically with empty body)

```
<html>
<head>
<title> Home Example </title>
    <meta charset = "utf-8">
    <meta name = "viewport" content = "width=device-width,
                                       initial-scale=1">
```

(CDN = content Delivery N/w)

go to the following path and copy the maxCDN code and paste here.

https://www.w3schools.com/bootstrap/bootstrap_get-started.asp

```
</head>
<body>
<div class = "container-fluid">
    <div class = "text-center">
        <h2><b> Hello {{name}} </b></h2>
    <h1>{{ content }}</h1>
    </div>
    <p> Thank you for using our website </p>
</div>
</body> </html>
```

Step 7 :-
go to urls.py file.

```
from django.conf.urls import url
from django.contrib import admin

from app2.views import home_page, contact_page, about_page

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^home/', home_page),
    url(r'^contact/', contact_page),
    url(r'^about/', about_page),
]
```

Step 8 :- Run the Server

```
(Pro2>python manage.py runserver
```

→ copy IP address → go to any browser → paste IP address → Enter.

→ we have to specify any of the four urls after the IP address in the url path. (/admin, /home, /contact, /about)

```
127.0.0.1:8000/home
Hello Dharsana
welcome to Home page
Thank you for using our website
```

5/7/18

Template Project 2 :-

→ develop a project to add two integer numbers given by user.

Step1 :- folder name : Adding numbers
Project name : addpro
App name : addapp

Step2 :- open Adding no. folder in pycharm

Step3 :- go to project.urls.py file and write the below code.

```
from django.conf.urls import url
from django.contrib import admin
from django.conf.url import include

urlpatterns = [
    url(r'^admin/', admin.site.url),
    url(r'^addapp/', include('addapp.urls'))
]
```

Step4 :- create urls.py file in the addapp application.

Addapp → right click → new → python file → urls.py (give the name)

step5:- go to addapp urls.py file and write the below code.

```
from django.conf.urls import url
from . import views
app_name = 'addapp'

urlpatterns = [
    url(r'^$', views.input, name='input'),
    url(r'^$', views.add, name='add'),
]
```

step6:- go to views.py file and write below code.

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

def input(request):
    return {render(request, 'addapp/base.html')

def add(request):
    x = int(request.GET['t1'])
    y = int(request.GET['t2'])
    z = (x+y)
    html = "The Addition of", "x", "and", y, "is", z
    return HttpResponse(html)
```

Step7:- click on adding numbers folder → new → Directory → Templates (give the name) → click o/k.

Step8:- click on ~~Templates~~ addapp folder → new → HTML → Base.html (give the name).

→ click on Templates folder → new → directory → addapp (give the name)

Step9:- open Base.html file and write below code.

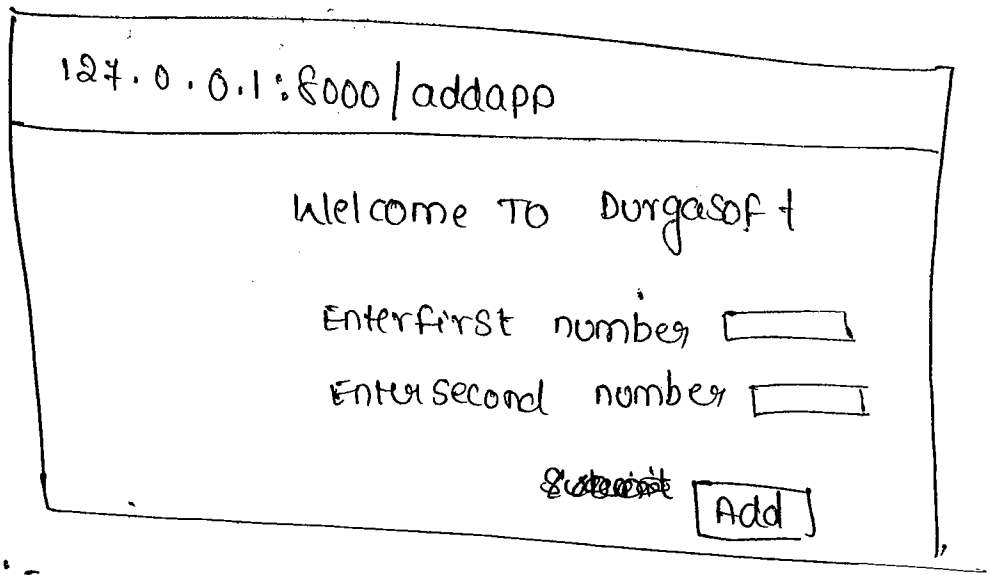
```
<html>
  <head>
    <title> input Page </title>
    <style>
      form{
      }
      h1 {
        Color : red ;
      }
    </style>
  </head>
  <body bgcolor= "# 00ffff" align = "Center">
<h1>welcom to Durgasoft</h1>
    <form action = "./add" method = "get">
      Enter first Number:- <input type= "text" name = "t1"><br>
      Enter second Number:- <input type = "text" name = "t2"><br>
      <input type = "submit" value= "Add">
```

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

step 10:- Run the Project

```
E:\Djangoprojects\Adding Two numbers> Python manage.py runserver
```

→ copy the IP address and Paste in Browser and specify \addapp after IP Address.

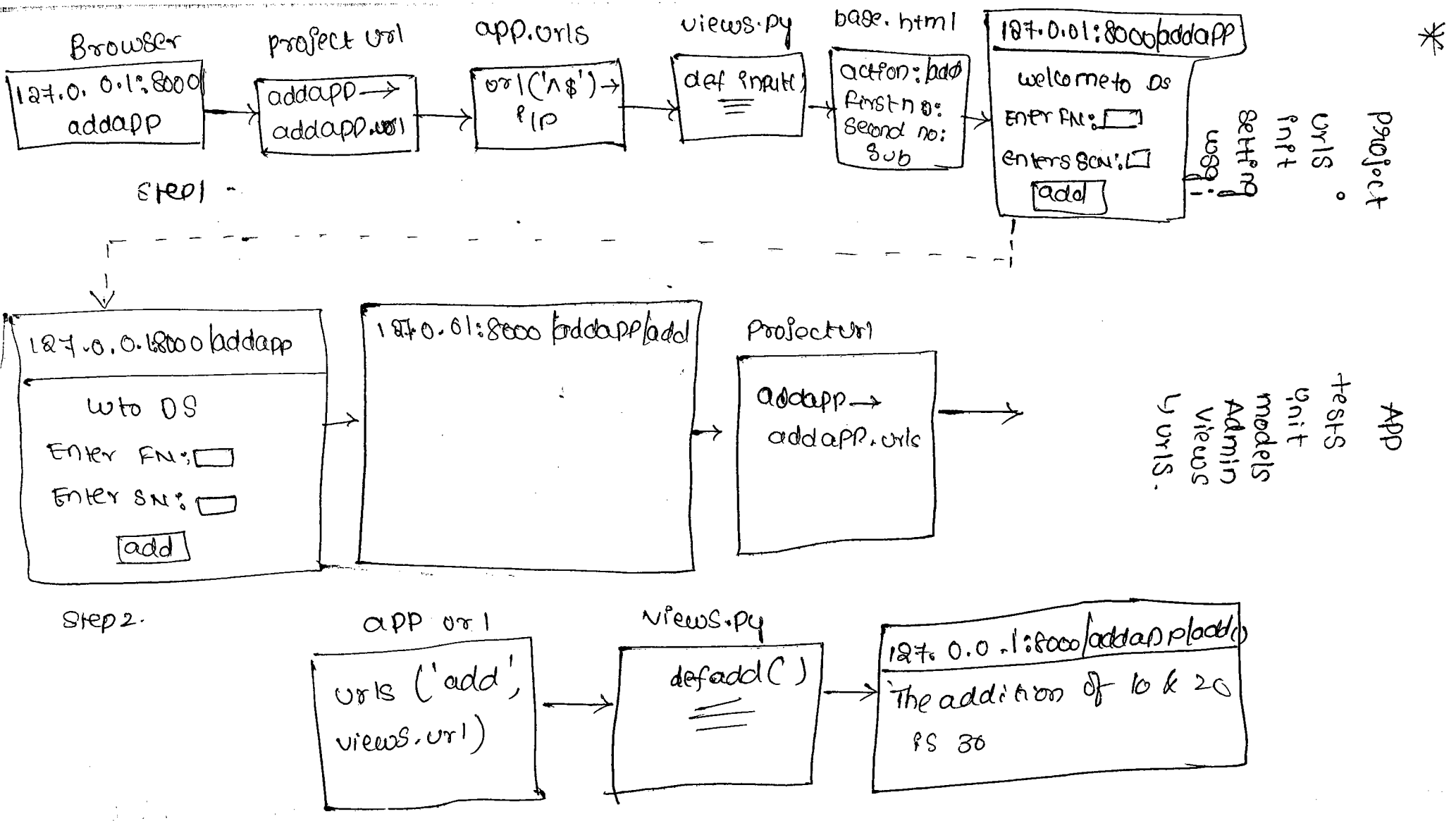


Step11:- now Enter First & second number and clicks on add button.

(10) (20)

```
127.0.0.1:8000/addapp/add?t1=10&t2=20
```

The addition of 10 and 20 is 30


Browser
127.0.0.1:8000
addapp
project url
addapp →
addapp.url
app.urls
url('^$') →
views.py
def input()
base.html
action: add
First no:
Second no:
Sub
127.0.01:8000/addapp
welcome to DS
Enter FN:
Enters SN:
add
Step1 -
127.0.0.1:8000/addapp
Wto DS
Enter FN:
Enter SN:
add
127.0.01:8000/addapp/add
Projecturl
addapp →
addapp.urls
Step2.
app url
urls ('add',
views.url)
views.py
defadd()
127.0.0.1:8000/addapp/add()
The addition of 10 & 20
is 30
project
urls
input
setting
app
tests
unit
models
Admin
views
urls.

Template project 3:- (Course Details)

courses :- Developed a project to display details of our courses.

Step I :- folder name :- template folder
project name :- pro1
App name :- app1

Step 2 :- open template in pycharm.

Step 3 :- go to urls.py file (main project) & write below code.

```
from django.conf.urls import urls
from django.contrib.import admin
from app1 import views
urlpatterns = [
    url (r'^admin|', admin.site.urls),
    url (r'^details|', views.cur_date),
]
```

Step 4 :- go to views.py file & write below code

```
from __future__ import unicode_literals.
from django.shortcuts import render
from django.http import HttpResponse
    def cur_date (request):
        return render (request, "welcome.
                                html")
```

Step 6:- create template folder then create html file with name welcome.html

```
<!Doctype html>
<html lang = "en">
<head>
    <meta charset = "UTF-8">
    <title> Durga soft </title>
</head>
<body>
<h1> welcome durgasoft </h1>
<h2> Batches in Durgasoft </h2>
<p> <b> List of Database </b> </p>
<ol>
    <li> oracle </li>
    <li> SQL server </li>
    <li> My SQL </li>
    <li> Post Grey SQL </li>
    <li> NOSQL </li>
</ol>
<p> List of language </p>
<ul>
    <li> Python </li>
    <li> Java </li>
    <li> .net </li>
    <li> C </li>
    <li> c++ </li>
</ul>
```

```
<h3> select Required framework </h3>
<Select>
    <option value = "Django"> Django </option>
    <option value = "flask"> Flask </option>
    <option value = "Google APP engine"> Google app
                                        engine </option>
</select>
<br>
  for Python Course content
<a href = "http://durgasoft.com/python-Naragana-
                                    mv2.asp"> Click Here </a>
</body>
</html>
```

Step 7 :- Run the server

Welcome Durga Soft

New Batch in Durgasoft

List of Database

1) Oracle
2. SQL Server
3. mysql
4. Postgreysql
5. NoSQL

List of languages

- Python
- Java
- C
- C++

Select required framework

Django

for Python course content click Here

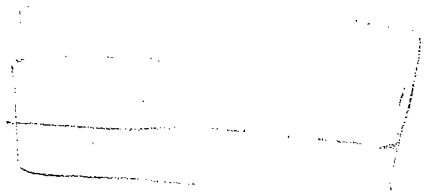
XS → mobiles → xtra small

containers → ① containers

body

POST Methode → take the data

Get → empty table



Forms :-

→ A form is a collection of Inbuilt classes and Libraries, all userdefined forms are derived from forms.form pre-defined form.

→ we have to create a new python file in the App with name 'forms.py'.

→ Creating a form is like creating a model.

→ whatever field types we use in the model Creation, the same field types are also used in the form creation.

→ all forms are automatically converts into code internally. and then that html code will create a form in the Browser.

step 1:- folder name : formfolder
project name : formpro
app name : formapp

step 2:- open formfolder in pycharm in pycharm.

step 3:- create a new python file and name it as forms.py.

step 4:- click on where formapp → new → python file → forms.py.

step5:- open forms.py then write the below code.

```
from django import forms
Class contactform (forms.form):
    fname = forms.charfield (max_length=220)
```

```
Lname = forms.charfield (max_length=20)
Loc = forms.charfield (max_length=20)
sal = forms.integerfield ()
email = forms.Emailfield (max_length=20)
```

step6:- go to command prompt then run the below command to go to python shell.

```
E:\formfolder\formpro> python manage.py shell
>> from formapp.forms import Contactform
>> c = Contactform()
>> print(c)
```

↓ It displays the HTML code regarding to contact form.

```
<tr>
    <th> <label for="id_fname"> Fname: </label> </th>
    <td> <input id="id_fname" maxlength="20" name="Fname"
                                    type="text" /> </td>
</tr>
<tr>
    <th> <label for="id_lname"> Lname: </label> </th>
    <td> <input id="id_lname" maxlength="20" name="Lname"
                                    type="text" /> </td>
</tr>
<tr> <th> <label for="id_loc"> loc: </label> </th>
    <td> <input id="id_loc" maxlength="20" name="loc"
                                    type="text" /> </td>
</tr>
```

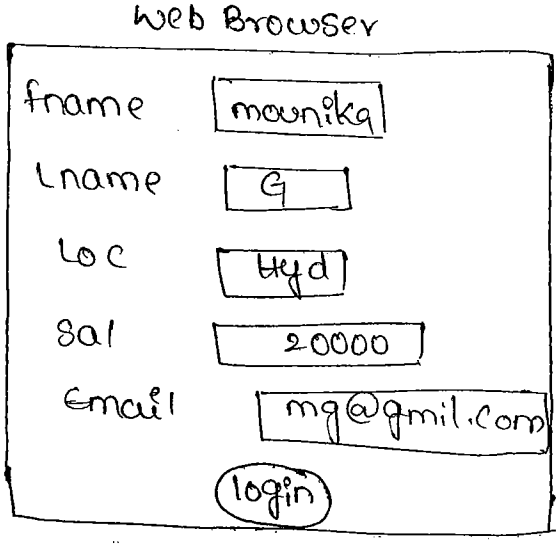
```
<tr>
    <th> <label for = "id_sal" > sal : </label> </th>
    <td> <input id = "id_sal" name = "sal" type = number" /> </td>
</tr>

<tr> <th> <label for = " id_email"> Email : </label> </th>
    <td> <input id = " id_email" name = "email" type = "email"/>
                                                    </td>
</tr>
```

(must be same)

tr → table row
th → table header
td → table data.

→ If the above code executes internally then a form will create like below.



* login button will be created manually

* is valid
↓ T
cleaned data
↳ {'fname' : 'mounika' ; 'lname' : 'G' ; 'loc' : 'Hyd' ;
'sal' : 20000 ; 'email' : 'mg@gmail.com'}

is_valid :

If the data entering in the text box is (in) correct pattern with out any errors then is valid is True.

↓(T)

cleaned_data :- cleaned Data convert all the Data in web browser in to dictonary format. (key : value)

{'fname': 'manika'; 'lname': 'g'; 'loc': 'hyd': 'sal': 20000; 'email': 'mg@gmail.com'}

↓

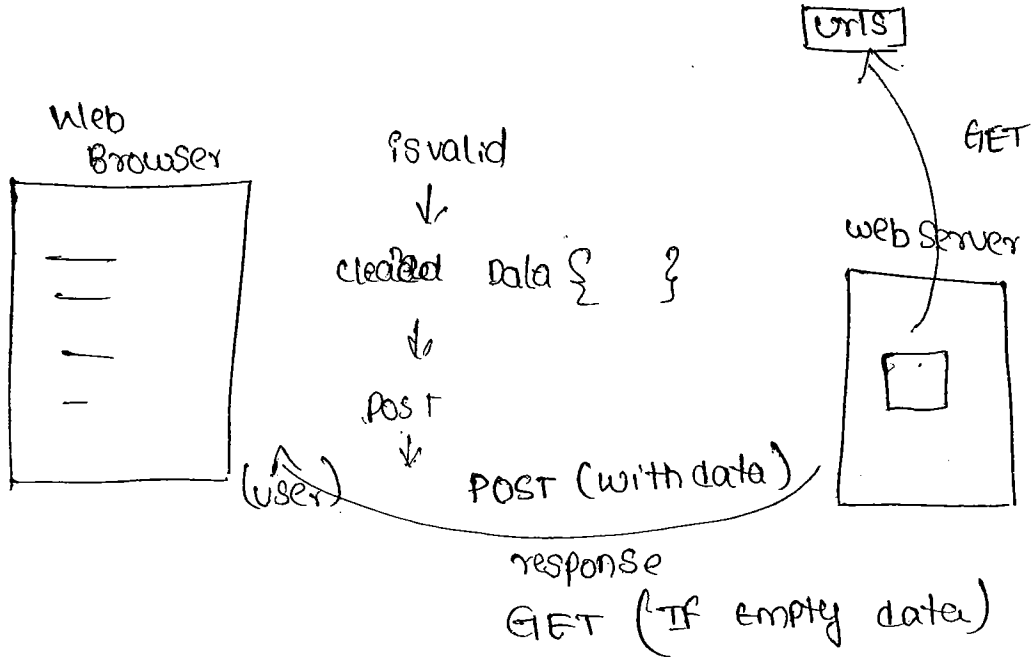
POST :-

which gives the all the Dict information to the webserver.

↓

webserver :- which gives the response to the user.

* and also GET methode used to take the data into urls.



Create Project to display contact form

STEP 1:-

folder name : form folder.

project name : form pro

app name: form app

STEP 2:- go to form.py

```
from django import forms

class contactform (forms.Form):
    fname = forms.CharField (label = 'First Name',
        Widget = forms.TextInput (
            attrs = {
                'class' : 'form-control',
                'placeholder' : 'Enter your first name'
            }
        )
    )
    lname = forms.CharField (label = 'Last Name',
        Widget = forms.TextInput (
            attrs = {
                'class' : 'forms-control',
                'placeholder' : 'Enter your last name'
            }
        )
    )
```

```
username = forms.CharField(label='User Name',
    widget = forms.TextInput(
        attrs = {
            'class' : 'form-control',
            'placeholder' : 'Enter your user name'
        }
    )
)

email = forms.EmailField(label = 'Email ID',
    widget = forms.EmailInput(
        attrs = {
            'class' : 'forms-control',
            'placeholder' : 'Enter your EmailID'
        }
    )
)

password = forms.CharField(label = 'password',
    widget = forms.password input(
                    ↓
                (If) Text input → it will be visible on Text box
        attrs = {
            'class' : 'form-control',
            'placeholder' : 'Enter your password'
        }
    )
)
```

```
mobile = forms.IntegerField (label = 'Mobile Number',
    Widget = forms.NumberInput (
        attrs = {
        (attribute)
            'class' : 'form-control',
            'Placeholder' : 'Enter your mobile number'
        }
    )
)
```

Step3:- Create html file and write the below code

```
<html>
<head>
<title> contact example </title>
```

go to www.w3schools ⟶ Bootstrap → Bootstrap get started and click on any ⟶ max CDN → copy the four lines <try yourself> → click on any try to yourself then copy the head section and paste here.

```
<style>
class .container {
        background-colour : antiquewhite;
    form {
        background-colour : Cadetblue;
```

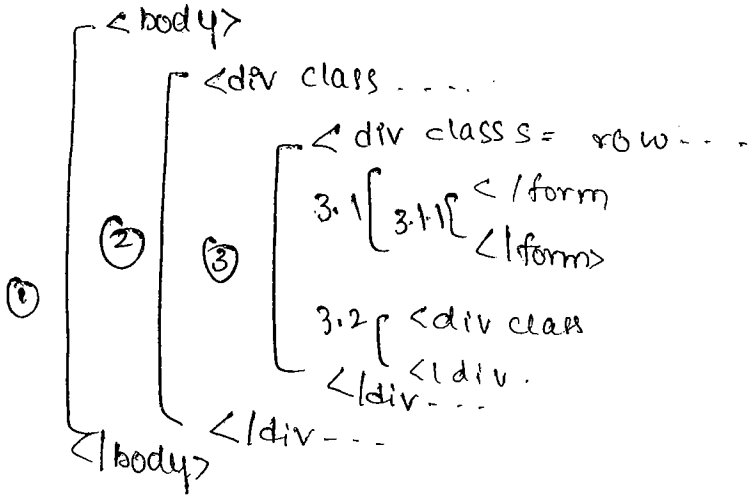
```
        border : 2px solid red ;
        border-radius : 5px ;
        padding : 5px 28px ;
        }
body {
        background-colour : aquamarine ;
        }
</style>
</head>
<body>
    <div class = "container">
    <h2> contact us </h2>
    <div class = "row">
        <div class = "col-md-4">
            <form method = "POST">
                { % . csrf_token % . }
            {{ form3 }} <br>
            <center><input type = "submit"
                           value = "submit"
                           class = "btn btn_default btn-md">
            </form>                                    </center>
        </ div>
            <div class = "col-sm-8">
            <h1><marquee behaviour = "alternate"
                                    scroll amount = "5">
                                    ↓
                                    respace
```

```
<h1><marquee behaviour= " . . . . . >
    welcome to Django
</marquee></h1>
</div>
</div>
</div>
</body>
</html>
```

Csrf
cross site request forgery
→ do not interrupt other person in same website

* ① Body
② container → for space
③ → row → 12 parts

column 1 — 4 parts → form
column 2 — 8 parts → msg ("welcome to django")



(3) a row contains generally 12 columns but here we used divided first four column as a one part i.e (3.1) and remaining 8 columns as a another part i.e 3.2

→ in 3.1 we create a form i.e 3.1.1 and in 3.2 we display a welcomehtle.

Step4:- go to views.py file write the below code.

```
from django.shortcuts import render
from django.http import HttpResponse
from formapp import forms
from formapp.forms import contactform

def index(request):
    if request.method = "POST":
        form = contactform(request.POST)
        if form.is_valid():
            return HttpResponse("Data successfully
                                 inserted")
        else:
            print(forms.errors)

    else:
        forms = contactform()
        return render(request, 'indexcontact.html',
                      {'form3': form})
```

Step5:-

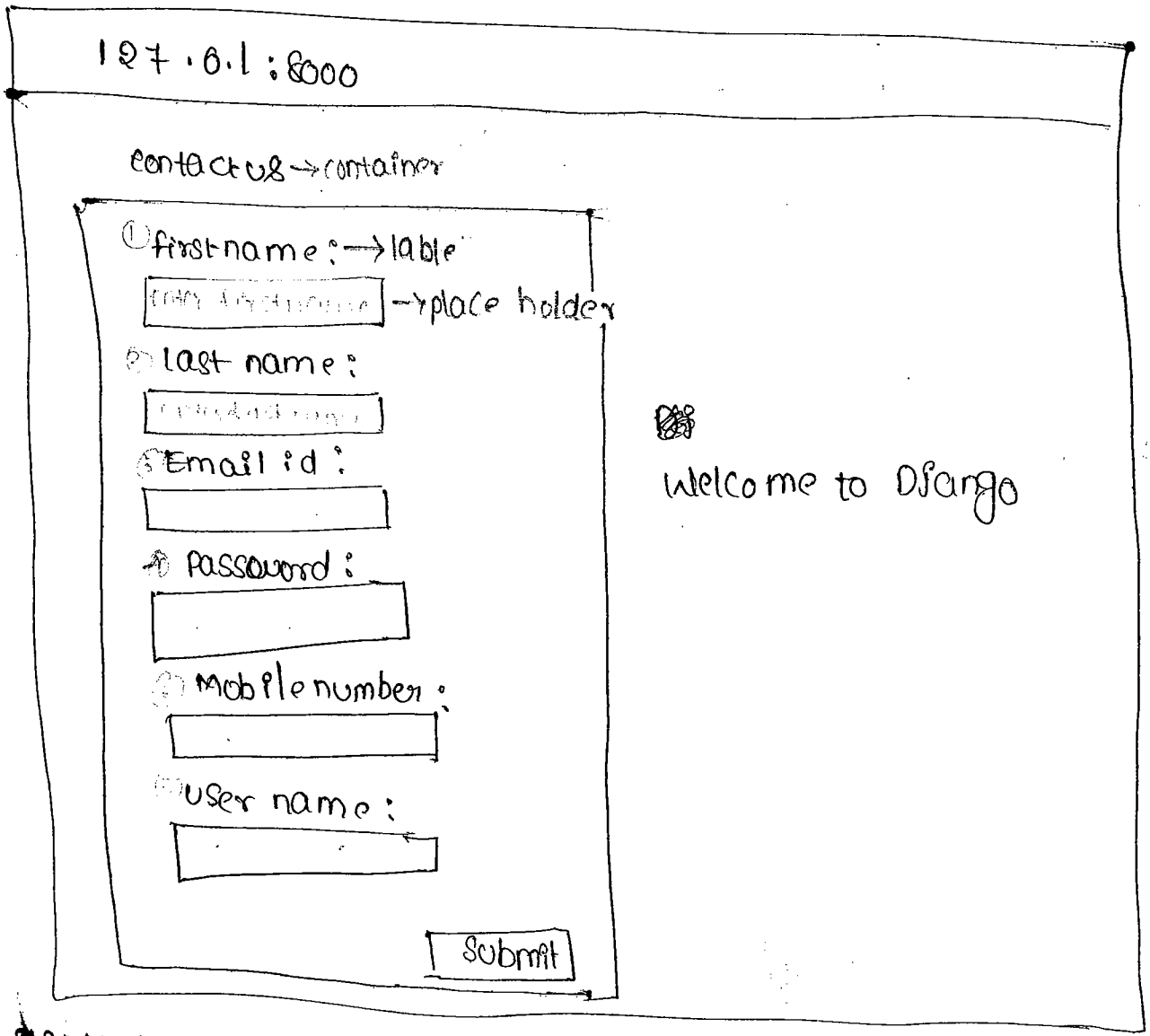
go to urls.py file

```
from django.conf.urls import url
from django.contrib import admin
from formapp import views
```

```
urlPatterns = [
    url(r'^admin/', admin.site.urls),
    (r'^$', views.index),
]
```

Steps:- Run the server

*go to browser → paste the Ip address → click Enter.



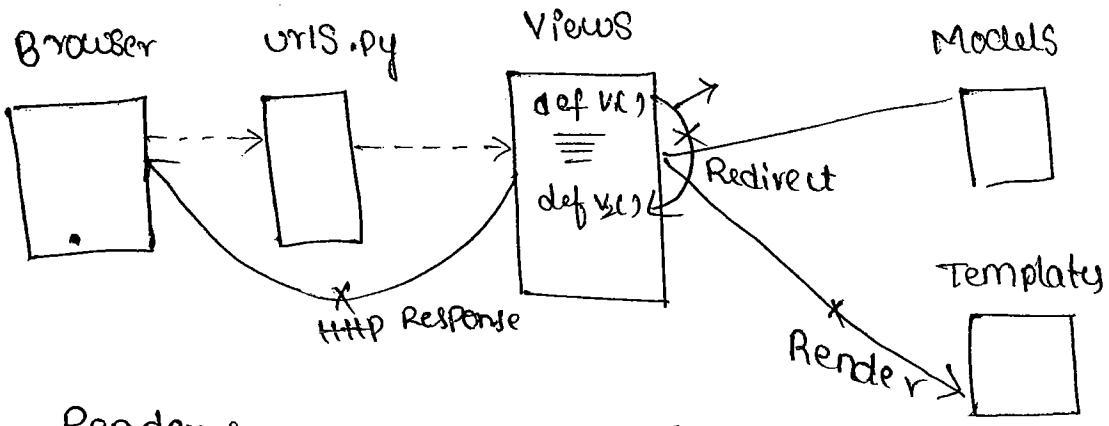
Step1:→ now we can give the data in all the fields & click submit

→Then it displays Response as

"Data successfully Inserted"

HttpResponse :-

It is used to Return & gives Response to Browser from views.



Render :-

It is used to transfer the Request from view to HTML file (template).

Redirect :-

It is used to transfer request from one view to another view.

① Diff Blw makemigrations and migrate

② Can we use only migrate without using makemigrations.

③ what is a class based view? → function based → def
→ classe based

④ What is the diff Blw function based views and class based views.

⑤ What is the defaultmanager in the model? object

⑥ why we use sessions in every website?

⑦ How to know that how many times the user same website?

⑧ what is the use of manage.py file?

⑨ what is the purpose of sqlmigrate command?

⑩ What is diff Blw Templates & forms?

⑪ When do we go for forms?

⑫ django framework follow MVT, T means template, then y template is not a default file in the djangoproject?

⑬ what are the different types of permissions to a user.

Active -

Staff Status -

Super " -

⑭ If we assign active and staff-status permissions to a new user, then what can he do in the website?

we have to members in the website, i dont want 3 users as a member of website, then what should we do?

(16) how to create a super user?

(17) create one user and P/w for MySQL database in one website, now i changed the database to Oracle, the can we use the same username and P/w for oracle also or do we need to recreate username and P/s?

Ans Yes we have to change both username & P/s (if we keep on changing database we need to change username & password)

(18) have you worked on html and CSS and JavaScript in the previous Project?

Yes, I involved a little bit.

(19) what is the purpose of abstract Model

In one Model, common fields are taken in to another Model.

(20) what is the Multiple inheritance, how is different from multiple inheritance.

Class Based views :-

Class = variable + method etc

→ Class Based views are an alternative way to implement views as a Python object instead of functions.

→ class Based views do not replace function Based views but they have certain differences & Advantages when compare to function Based views.

→ A view is a callable which takes a request and returns a response.

views → Python function
↳ also use class Method

→ This can be more than a function.

→ Django provides some classes which can be used as a views. this allow us to structure our views by using Inheritance. i.e we can use the code.

step 1 :- folder Name : ClassBasedVies
Project name : ClassBasedPro
Application name : genericviews

step 2 :- Open ClassbasedFolder in Pycharm

↓

go to project level urls and write the below code.

```
from django.conf.urls import url
from django.contrib import admin
from django.conf.urls import include

urlpatterns = [
    url(r'^admin/', admin.site.urls),
```

→ If we specify genericviews in the Browser Path, ' then it will go application level urls.

→ If we don't specify genericviews in the url path then it will check for '^$' (empty) in Project level urls.py.

But In project level urls doesnt hava '^$' So it will return 'Page not found error'

→ In Project level urls if we specified genericviews to map application level urlsfile thats why we should specify genericviews in url path after IP address in all the classes.

127.0.0.1:8000/genericviews

Step3:- goto genericviews app

→ genericviews (Righ click) → new → Python file → urls.py

Open app level urls.py file and write the below code.

```
from django.conf.urls import url
from genericviews import views
app_name = 'genericviews'
urlpatterns = [
```

class Based (as_view)

```
    url(r'^$', views.indexviews.as_view(), name='index'),
    url(r'^(?P<pk>[0-9]+)|$', views.Detail&view.as_view(),
                                    name='detail'),
```

↓ variablename (anyname)

url (r'^makeentry$', views.makeentry, name='makeentry'):

(function Based)

Step4:- go to views.py file

```
from django.shortcuts import render
from django.views import generic
from genericviews.forms import PersonForm
from genericviews.models import Person

def makeentry(request):
    if request.method == 'POST':
        form = PersonForm(request, POST)
        if form.is_valid():
            name = request.POST.get('name', '')
            desc = request.POST.get('desc', '')
        person = Person(name = name, desc = desc)
        person.save()
        form = PersonForm()
        return render(request, 'genericviews/makeentry.html',
                      {'form': form})
                       key  value
    else:
        form = PersonForm()
        return render(request, 'genericviews/makeentry.
                      html', {'form': form})
```

```
class indexview (generic.listview):
    context_object_name = 'list'
        ↳ variable name (any)
    template_name = 'genericviews/index.html'

    def get_queryset (self):
            ↳ default → to display one by one
        return person.objects.all ()

class DetailsView (generic.Detailview):
    model = Person
    template_name = 'genericviews/detail.html'
```

STEP 5 :-

→ create templatesfolder

→ create html file with name makeentry.html file.

```
<!DOCTYPE html>
<html lang = "en">
 <head>
```

[copy files from internet]

```
  <style>
      body {
          background-color : paleturquoise;
      }
class ← .container
          background-color : antiquewhite;
```

```
        border-radius : 10px ;
        margin-top    : 40px ;
    }
    form {
        border : 5px ;
        border-color : red ;
    }
</style>
</head>
<body>
<div class = "container">
    <h1 class = "text-center"> Please make entries here</h1>
    <div class = "col-md-5">
        <form action = "{% url 'genericviews:makeentry' %}
              methode = "post">
            {% csrf_token %}
            {{form}} <br>
            <center><input type = "submit"
                    value = "submit"></center>
        </form>
    </div>
</div>
<body>
</html>
```

step6:- create an html file with name index.html

```
{% if list %}
    <h1> Person Details ! </h1>
    <ul>
        {% for person in list %}
            <li>
                <a href = "{% url 'generviews:details'
                    person.pd %}"> {{
                person.name }} </a>
            </li>
        {% endfor %}
    </ul>
{% else %}
<h3> No product found. </h3>
{% endif %}
```

step7:- Create html file with name details.html

```
<h1> {{ person.name }} </h1>
<h3> {{ person.desc }} </h3>
```

{{ }} variable

step8:- go to models.py file write below code.

```
from django.db import models
class person (models.Model):
    name = models.charfield (max_length=100)
```

```
    desc = models.CharField(max_length=300)
    def __str__(self):
        return self.name
```

step 4: Create forms.py file

```
from django import forms
class PersonForm(forms.Form):
    name = forms.CharField(
        widget = forms.TextInput(
            attrs = {
                'class': 'form-control',
                'placeholder': 'Enter name',
            }
        )
    )
desc = forms.CharField(help_text = "write a brief about the person.",
    widget = forms.Textarea(
        attrs = {
            'class': 'forms-control',
            'placeholder': 'Enter description',
        }
    )
)
```

Step 10:- Create a database with name genericdb and configure the database in settings.py file.

```
DATABASE = {
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
        'NAME' : 'genericdb',
        'USER' : 'root',
        'PASSWORD' : 'root',
    }
}
```

Step 11:- Configure html files in settings.py file.

```
TEMPLATES = {
    {
    'BACKEND' : 'django.template.backends.django.DjangoTemplate',
    'DIRS' : [ os.path.join (BASE_DIR, 'templates')]
```

Step12:- Run the server

```
> python manage.py runserver
```

→ Copy IP Address & paste In browser along with /genericviews.

127.0.0.1:8000/genericviews.

↓(Enter) ↳ go to project level urls.

No person found.

→ Initially there are no persons in the database thats why it displayed no persons found.

→ now we will specify the data for that use makeentry.

127.0.0.1:8000/genericviews/makeentry

↓(Enter)

Please make entries here

Name : Nani

Description : Hello Nani

write a brief about the person.

Submit

→ again enter one more detail

Name : Mouni

description: Hai Mouni

Submit

→ To see list of persons only use genericviews

127.0.0.1:8000/genericviews

Person Details :

- Nani
- Mouni

→ If we click on Nani name then it shows auto-matecall id as 1 in Browser.

127.0.0.1:8000/genericviews/1

Nani

Hello Nani

Registration and login page :-

Step 1 :-

foldername : loginfolder

Projectname : login

Application name : regandloginapp

Step 2 :- go to loginfolder in pycharm

go to models.py file and write below code.

```
from __future__ import unicode_literals
from django.db import models

class Reg(models.Model):
    fname = models.CharField(max_length = 20)
    lname = models.CharField(max_length = 20)
    user = models.CharField(primary_key = True, max_length
    pwld = models.CharField(max_length = 20)                =20)
    mobile = models.CharField(max_length = 20, unique = True)
    email = models.EmailFied(max_length = 20, unique =
    dob = models.DateField(null = True)                    True)
    gender = models.CharField(max_length = 20, blank = True,
                                                null = True)
```

Step 3:- go to forms.py file <App level>

```
from django import forms
from .models import Reg  -> (model name)
from django import forms

class RegForm (forms.Form):
    fname = forms.CharField(
        widget = forms.TextInput(
            attrs = {
                'class' : 'form-control',
                'placeholder' : 'Enter First Name'
            }
        )
    )

    lname = forms.CharField(
        widget = forms.TextInput(
            attrs = {
                'class' : 'form-control',
                'placeholder' : 'Enter last name'
            }
        )
    )
```

```
pwid = forms.CharField(
    widget = forms.PasswordInput(
        attrs = {
            'class': 'form-control',
            'placeholder': 'Enter User Name'
        }
    )
)

mobile = forms.CharField(
    widget = forms.NumberInput(
        attrs = {
            'class': 'form-control',
            'placeholder': 'Enter us Mobile number'
        }
    )
)

email = forms.CharField(
    widget = forms.EmailInput(
        attrs = {
            'class': 'form-control',
            'placeholder': 'Enter emailid'
        }
    )
)
```

< Before this user name missing >

gender =

dob = forms.CharField(

widget = forms.TextInput(

attrs = {

'class': 'form-control',

'placeholder': 'Enter your Date of Birth'

}

)

)

gender = forms.CharField(

widget = forms.TextInput(

attrs = {

'class': 'form-control',

'placeholder': 'Enter your gender'

}

*for login (forms)

class loginform(forms.Form):

user = forms.CharField(max_length=20)

PWd = forms.CharField(widget=forms.PasswordInput())

step4: go to urls.py file in projectlevel

```
from django.conf.urls import url
from django.urls import admin
from django.conf.urls import include

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^regandloginapp/', include('regandloginapp.urls')),
```

Step5:- go to urls.py file in applevel

app → Right click → new → Python file → urls.py (write)

```
from django.conf.urls import url
from. import views

app_name = 'loginapp'
            ↳ 'regandloginapp'

urlpatterns = [
    url(r'^$', views.home, name='home'),
    url(r'^reg$', viewo.logon
                  reg, name='reg'),
    url(r'^loginp$', views.login, name='login'),
]
```

Step6:- go to views.py file.

```
from django.shortcuts import render
from django.http import HttpResponse Redirect, HttpResponse
From.models import Reg
                   ↳ model name
from.forms import loginform, RegForm

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

```
def reg (request ):
    if request.method == 'POST':
        form = RegForm(request. POST)
        if form. is_valid ():
            fname = request. POST.get ('fname', '')
            lname = request. POST. get ('lname', '')
            user = request. POST. get ('user', '')
            Pwd = request. POST. get ('pwd', '')
            mobile = request. POST. get ('mobile', '')
            email = request. POST. get ('email', '',
            dob = request. POST. get ('dob', '')
            gender = request. POST. get ('gender', '')
        reg = Reg(fname = fname, lname = lname, user = user,
                  Pwd = Pwd, mobile = mobile, email = email,
                  dob = dob, gender = gender)
        reg. save()
        return HttpResponse ('reg success')
    else :
        form = RegForm()
        return render (request, 'reg.html', {'form': form})

def login (request ):
    if request. method == "POST" :
        Myloginform = loginForm (request. POST)
        if Myloginform. is_valid ():
            user = Myloginform. cleaned_data ['user']
```

```
        pwd = myloginform.cleaned_data ['pwd']
        dbuser = Reg.objects.filter (user = user)
        dbpwd = Reg.objects.filter (pwd = pwd)
        if not dbuser or dbpwd:
            return HttpResponse ('login success')
        else:
            return HttpResponse ('login failed')
    else:
        form = loginform()
        return render (request, 'login.html', {'form': form}
```

STEP 7: Create a template folder in the Project folder whereever manage file is there.

→ login → new → directories → template

→ click on templates folder → new → htmlfile → home.html (give manually)

*Open home.html file and write the below code.

```
<html>
<head>
    <style>
        div {
            padding-top: 20px;
        }
        i {
            color: red;
        }
        h1 {
            color: green;
        }
        #id1 {
            padding-top: 130 px;
            font-size: x-large;
```

```
    }
  </style>
</head>
<body bgcolor = "#00ffff">
<div align = "center">
      <h1 class = "h1"> welcome to <i> Django </i> world</h1>
      <h2><a href= "./reg"> click here to register</a></h2>
      <h2><a href= "./login"> click here to login </a></h2>
 </div>
<pre id = "id1" style = "float :right ;">
      <b> By </b>
       Narayana,
       +91-9010607010
       Pythonnarayana@gmail.com
       Durga Soft,
       Mitrivanam,
        Hyderabad.
 </pre>
 </body>
 </html>
```

steps: Create html file with name reg.html in templates folder

R.C → template → new → reg.html
→ Html

```
<html>
<head>
<title> welcom to registration </title>
```

{ from w3schools.com
↳ cpy → pst

```
<style>
    body { background-color: aque; }
     b{
       font-size : 25px;
      }
      form {
          background-color : pale turquoise;
           border : 5px solid red;
           border-radius : 5px;
            Padding : 10px 10px;
       }
       # htag {
            color : red;
       }
    </style>
</body>
<h1 class = "text-center" id = "htag"> welcom to Django website
                                                    </h1>
<div class = "container">
    <h4> please register your details here </h4>
    <div class = "row">
        <div class = "col1-md-4">
            <form method = "POST">
                {% csrf_token %}
                {{ form }} <br>
                <div class = "text-center">
                 <input type = "submit"
                        value = "submit"
                        class = "btn btn-default btn-md">
```

```
<input type = "reset"
       value = "Reset"
       class = "btn btn-default btn-md">
    </div>
  </form>
 </div>
<div class = "col-sm-8">
    <h4 align = "center"> <a href = "./"> Goto Home Page
                                    </a></h4>
 </div>
</div>
</div>
</form>
</body>
</html>
```

Step9 create html file with name login.html in templates folder.

step9:- Template folder → R.C → new → login.html

→ open login.html and write below code.

```
<html>
<head>
    <style>
        table {
            margin-top : 30px;
        }
    </style>
</head>
```

<body bgcolor=aqua>

<h3 align="center"> please enter your details to logfn </h3>

<form action="" method="POST">

{% csrf_token %} → tag for security (it do not disturb others (cross site request forgery))

<table bgcolor="#deb887" align="center">

<tr><td> User Name : </td><td>{{ form.user }}</td></tr>

<tr><td> Password : </td><td> {{ form.pwd }} </td></tr>

<tr><td></td> <td><input type="submit" value="log fn"> </td></tr>

</table>
 → break

<h2 align="center"> Go to Home page </h2>

<h2 align="center"> Go to Registration page </h2>

</body> → </form>

step 10:- go to mysql and create database with name

```
mysql> create database dbadmin
```

use dbadmin ;

show tables ;

↓

empty data

step 11:- go to settings.py file and write below

```
DATABASE = {
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
        'NAME' : 'dbadmin',
        'USER' : 'root',
        'PASSWORD' : 'root',
```

Step 12:- Configure Templates in template option in settings.

```
TEMPLATES = [
    {
        'BACKEND':
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        'APP_DIRS': True,
```

Step 13:- Run makemigration command

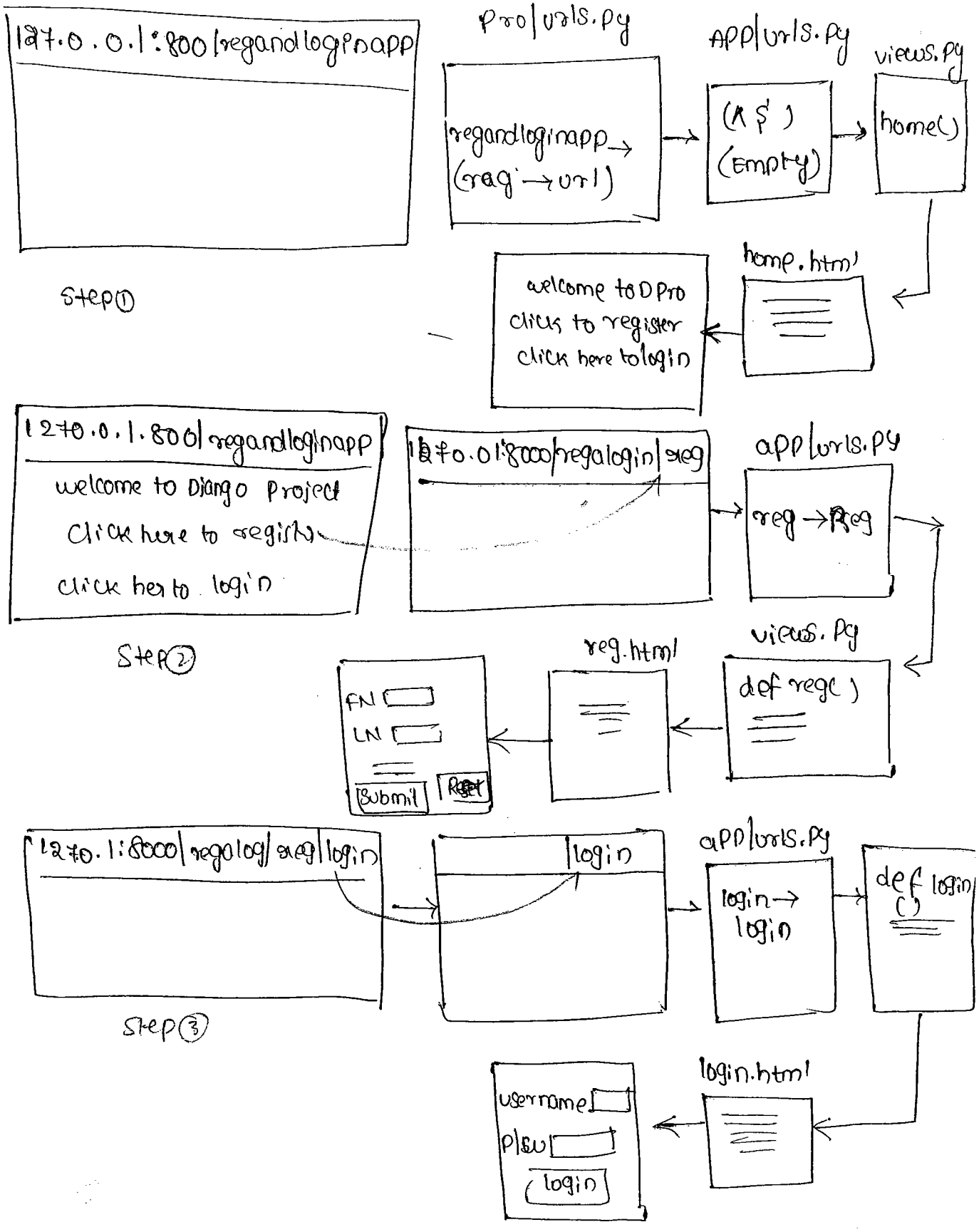
```
> python manage.py makemigrations
```

Step 14:- Run migrate command.

```
> python manage.py migrate
```

Step 15:- Run server command

```
> python manage.py runserver
```


127.0.0.1:800/regandloginapp
pro/urls.py
regandloginapp→
(reg→url)
APP/urls.py
('')
(Empty)
views.py
home()
home.html
welcome to DPro
click to register
click here to login
Step①
127.0.0.1:8000/regandloginapp
welcome to Django Project
Click here to regisTr
click her to login
127.0.01:8000/regalogin/reg
app/urls.py
reg→Reg
views.py
def reg( )
reg.html
FN
LN
Submit
Reset
Step②
127.0.1:8000/regalog/reg/login
login
app/urls.py
login→
login
def login
( )
login.html
username
PSW
login
Step③

① FH+HB

Create a project to store form data into database by using 'GET' Methode

step1 :- folder name : getfolder

Project name : getpro

app name : getapp

step2 :- open getfolder in pycharm

→ go to models.py file

```
from django.db import models
class getmodel (models.Model):
    name = models.CharField (max_length = 100)
    job = models.CharField (max_length = 100)
    message = models.CharField (max_length = 100)
    created_at = models.DateTimeField()
```

step3 :-

create a forms.py file & write below code

app → r.c → Python file → forms.py

```
from django. import forms
class getform (forms.Form):
    name = forms.CharField (max_length = 100,
        widget = forms.TextInput (
            attrs = {
                'class' : 'form-control',
                'place holder' : 'enter your name'
            }
        )
    ,
```

```
job = forms. CharField ( max_length =100,
widget  = forms. TextInput (
    attrs = {
        'class': 'form-control',
        'place holder': 'enter your job',
        }
    )
)
message = forms. CharField ( max_length = 206,
    widget = forms. TextInput (
        attrs = {
            'class': 'forms_control',
            'placeholder': 'enter message'
            }
        )
    )
Created_at = forms. DateTimeField (
    widget = forms. DateTimeInput (
        attrs = {
            'class' : 'form_control',
            'placeholder': 'enter date'
            }
        )
    )
```

Step 4:- go to urls.py file

```
from django. conf. urls import url
from django. contrib import admin
from getapp import views
```

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^getfile.html$', 'getapp.views.getview')
]
```

step5:- go to views.py file

```
from django.shortcuts import render
from getapp.forms import getform
from .models import getmodel
from django.http import HttpResponse
def getview(request):
    if request.method = 'GET':
        form = getform()
    else:
        form = getform(request.POST)
        if form.is_valid():
            name = form.cleaned_data['name']
            job = form.cleaned_data['job']
            message = form.cleaned_data['message']
            created_at = form.cleaned_data['created_at']
getmod = getmodel.objects.create(name=name, job=job,
                                 message=message,
                                 created_at=created_at)
    return HttpResponse('Thanks you')
    return render(request, 'getapp/getfile.html',
                  {'form': form})
```

Step 6 :-

Create html file with name getfile.html

(getpro → Template → gethtml)

```
<form action = '/getfile.html' method= 'POST'>
        {% csrf_token %}
        {{ form.as_p}}
        <input type = 'submit' value = 'submit' />
        </form>
```

STEP 7 :-

→ Create a database with name getdb

→ Create database getdb;

→ use getdb;

STEP 8 :-

Go to settings.py file and configure the database

```
DATABASE = {
    'default' : {
        'ENGINE' : 'django.db.backends.mysql',
        'NAME' : 'getdb',
        'USER' : 'root',
        'PASSWORD' : 'root',
        }
    }
```

Step 9:- Run server

```
python manage.py runserver
```

paste IP address in Browser along with /getfile.html

127.0.0.1:8000/getfile.html

Name : enter your name

Job : enter your Job

Message : enter message

created_at : date

Submit

Enter data

Name : venkat

Job : Polic

message : Im Polic officer

created_at : 2018-10-10

Submit

↑ click on submit

* Click on submit button

* Thank you, message display

Step 10 :-

go to database & check the data

select * from getAPP_getmodel;

id	name	Job	message	created-at
1	venkat	Polic	Im Police officer	2018-10-10

* we should run makemigrations & migrate command before running server.

Page redirection :-

→ Page redirection is a common task in every website.

→ if we don't redirect the request or user in the website then the website can't work properly.

→ for example, if user enter username and plw in the login page then we have to redirect the user to the home page any required page as per the website.

→ we use redirect function to redirect the user blw two pages (or) two tasks in the same page.

→ import redirect from django.shortcuts. now we will implement a small project to perform Mathematical operations. (Add, Sub, mu, div)

NOTE :- We can redirect one view to another view (same page)
We can redirect one view to the url (one page to another page)

Step 1 :- foldername : calculationfolder
Project name : calculationpro
Application name : calculationapp

Step 2 :- open Pycharm → calculationfolder
→ go to urls.py file

```
from django.contrib import admin
from django.conf.urls import url
from calculationapp import views

urlpatterns = [
    url(r'admin/', admin.site.urls),
    url(r'^$', views.input, name='input_def'),
    url(r'^calculationapp/output$', views.output, name='output_def'),
```

```
url( r'^add$', views.add),
url (r'^sub$', views.sub),
url (r'^mul$', views.mul),
url (r'^div$', views.div),
]
```

step3:- go to views.py file and write below code.

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
    i = None
    j = None

def input (request):
        return render (request, "add.html")

def output (request):
        val1 = request.GET['t1']
        val2 = request.GET['t2']
        global i
        global j
        i = int(val1)        int(int(val1))
        j = int(val2)
        z = request.GET['out']
        if z == 'add':
            return redirect (add) # go to add view
        if z == 'sub':
            return redirect (sub) # go to subview
        if z == 'mul':
            return redirect ('mul') # go to mulview
        if z == 'div':
            return redirect ('/div')
```

go to url path → views (div)

```
def add(request):
    k = i+j
data = "addition of", i, "and", j, "is", k
return HttpResponse(data)

def sub(request):
    k = i-j
data = "subtraction of", i, "and", j, "is", k
return HttpResponse(data)

def mul(request):
    k = i*j
dat = "multiplication of", i, "and", j, "is", k
return HttpResponse(data)

def div(request):
    k = i/j
data = "division of", i, "and", j, "is", k
return HttpResponse(data)
```

STEP 4 :- create html file with name add.html

```
<from action = "./calculation_app/output" methode "get">
    {% csrf_token %}
    Enter fno : <input type = "text" name = "t1"><br><br>
    Enter sno : <input type = "text" name = "t2"><br><br>
                <input type = "submit" value = "add" name = "but">
                <input type = "submit" value = "sub" name = "but">
                <input type = "submit" value = "mul" name = "but">
                <input type = "submit" value = "div" name = "but">
</form>
```

Step5 :- go to settings.py file and configure the Installed APP in TEMPLATES option.

Step6: Run the Server

→ copy the IP address & paste in Browser

127.0.0.1:8000

Enter fno : [10]

Enter sno : [5]

[add] [sub] [mul] [div]

* If we click on add button → o/p is

The addition of 10 and 5 is 15

* if we click on sub button → o/p is

The subtraction of 10 and 5 is 5

* if we click on mul button → o/p is

The multiplication of 10 and 5 is 50

* if we click on div button → o/p is

The division of 10 and 5 is 2

30/7/18

MIME :- (Multipurpose Internet Mail Extension)

Step1 :- Project name : MIME_APP

folder name : MIME_folder

APP name : MIME_APP

Step2 :- go to urls.py file

```
from django.contrib import admin
from django.conf.urls import url
from MIME_APP import views
```

```
from MIME_App import views

urlpatterns = [
    url('admin/', admin.site.urls),
    url('mimeapp/html', views.html_page),
    url('mimeapp/xml', views.xml_page),
    url('mimeapp/json', views.json_page),
    url('mimeapp/word', views.word_page),
    url('mimeapp/excel', views.excel_page),
```

Step3:-

go to views.py file.

```
from django shortcuts import render
from django.shortcuts import HttpResponse
def html_page(request):
    data = "<html><body><table border='1'>"
           "<tr><th>ID</th><th>Name</th><th>marks</th>
           "<tr><td>101</td><td>srinu</td><td>90</td></tr>"
           "<tr><td>102</td><td>vamsi</td><td>95</td></tr>"
           "<tr><td>103</td><td>narayana</td><td>100</td></tr>"
           "</table></body></html>"
    return HttpResponse(data, content_type="application/html")
def xml_page(request):
    data = ( }//same code
    return HttpResponse(data, content_type="application/xml")
def json_page(request):
    data = ( } // same code
    return HttpResponse(data, content_type="application/json")
```

→ def word_page(request):
    data = {≡} // same code
    return HttpResponse(data, Content type = 'application | msword')

→ def excel_page(request):
    data = (≡) // same code
    return HttpResponse(data, Content type = 'application | excel')

Step4:-

① Run makemigrations
② Run migrate
③ Run server

Step5:- go to Browser paste IP address < it specify all the possibilities > Eg: html, excel, word, JSON shows

Starting:- 127.0.0.1:8000 | — empty

Next:- enter any possibility manually 127.0.0.1:8000 html

ID	Name	Marks
101	Srinu	90
102	vamsi	95
103	Narayana	100

NOTE:- ① if we specify ~~any~~ mimeapp|xml then it display the same data in xml formate.