

INSTALLING DJANGO

It is a best practice to always create your folders and files anywhere other than in your C drive.

C:\Users\Ummesalma>F:

Create a folder called 00Django

F:\>mkdir 00Django

F:\>cd 00Django

VIRTUAL ENVIRONMENT

Virtualenv: Before installing Django, it's recommended to install Virtual environment because it creates a new virtual space isolates your Python files and softwares from other projects. This will also ensure that any changes made to your current project won't affect other projects/websites you're developing.

Create a virtual environment so that if at all you need to update any of the packages it won't disturb your other projects.

The syntax used for creating a virtual environment in python is very simple.

python -m venv <Environment name>

Here, venv is a keyword.

F:\00Django>python -m venv MYENV

After creating a virtual environment, it is important to activate it then only we can use it.

F:\00Django>MYENV\Scripts\activate

Once the virtual environment is activated you can observe your given environment before the command prompt.

(MYENV) F:\00Django>pip install Django

Note: pip is used in python for installation. If the new version of pip is available, you will be shown a message regarding this and recommend you to update pip. If you want to update pip then use the following command.

python -m pip install --upgrade pip

You can check the version of your Django by giving

python -m django --version

IT IS THE TIME TO CREATE YOUR 1st DJANGO PROJECT

In order to create a project you need to give the following command

(MYENV) F:\00Django>django-admin startproject projectDemo

Go to project. Inside the project you will find **manage.py** file which is our master file for our further actions. Along with it a folder with the same name as that of the project name will be found (here projectDemo). Inside this folder we have 5 core files named `__init__.py`, `asgi.py`, `wsgi.py`, `urls.py` and `settings.py`

`__init__.py` (which acts as a constructor), `asgi.py` and `wsgi.py` files are there for providing the compatibility between web servers, frameworks, and applications. `urls.py` is used for managing the URLs. And `settings.py` is a core file in Django projects. It holds all the configuration values that your web app needs to work. The settings may be related to login setting, database settings and many more.

In order to make the project executable we need to migrate into the project environment where all the required files, folders and databases are auto created. In order to migrate into project environment one should give the following code:

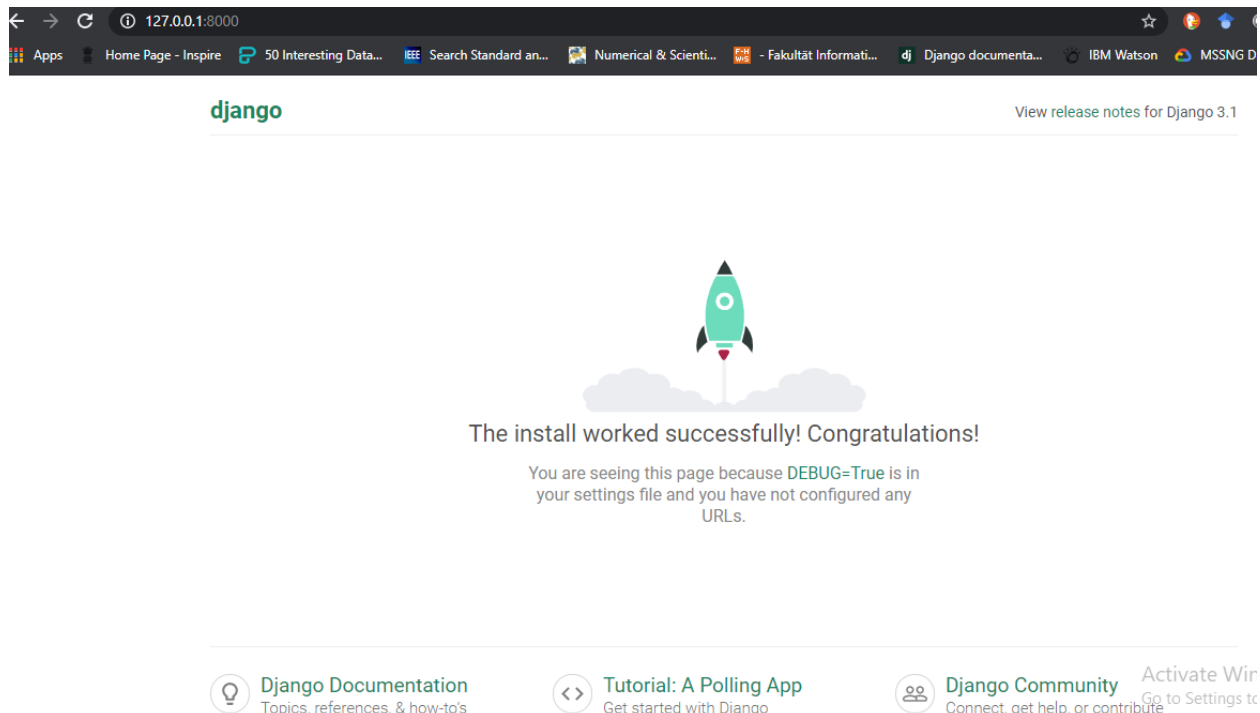
(MYENV) F:\00Django>django-admin startproject projectDemo

(MYENV) F:\00Django>cd projectDemo

(MYENV) F:\00Django\projectDemo>python manage.py migrate

Now run the server to check migration has happened successfully

(MYENV) F:\00Django\projectDemo>python manage.py runserver



Once you migrate into the working environment and server is running properly it is required to authenticate the credentials. i.e. you need to provide username, password and email_id details. Since it is your project you will be the super user thus give the following command to authenticate superuser.

(MYENV) F:\00Django\projectDemo>python manage.py createsuperuser

Username (leave blank to use 'ummesalma'):

Email address: hmbssalma@gmail.com

Password: *****

Password (again): *****

Superuser created successfully.

Once the super user is created successfully go to your web browser and check the starting page.

Type server address **127.0.0.1:8000**

To confirm the admin credentials type 127.0.0.1:8000/admin

You will be asked to confirm the username and password. And you are done with your empty project creation and ready to go for building your 1st app.

BUILDING THE 1st APP INSIDE THE PROJECT

Open any editor (here I am using sublime as it provides side panel for viewing the contents of the folder.)

Drag and drop the project into the sublime. While dragging you should remember that the project (here projectDemo) in your drive from where you are dragging should be at the same level as that of your virtual environment.

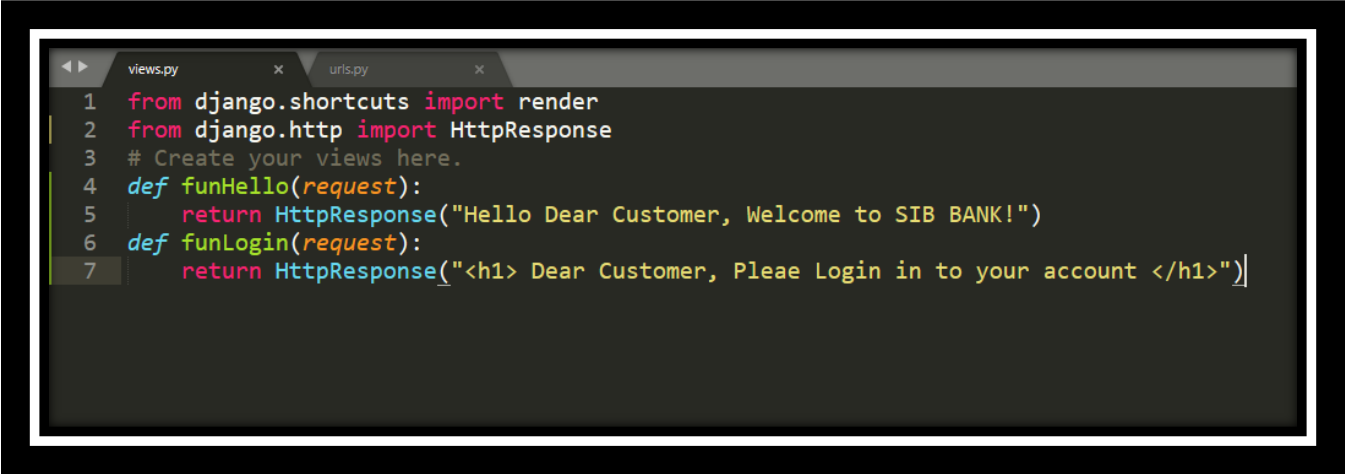
Now press control+break in your cmd prompt and type

(MYENV) F:\00Django\projectDemo>python manage.py startapp appDemo

Go to sublime you can see that a folder called appDemo is created. Click on appDemo folder. You can observe some prebuild files which include admin.py, apps.py, models.py, tests.py and view.py.

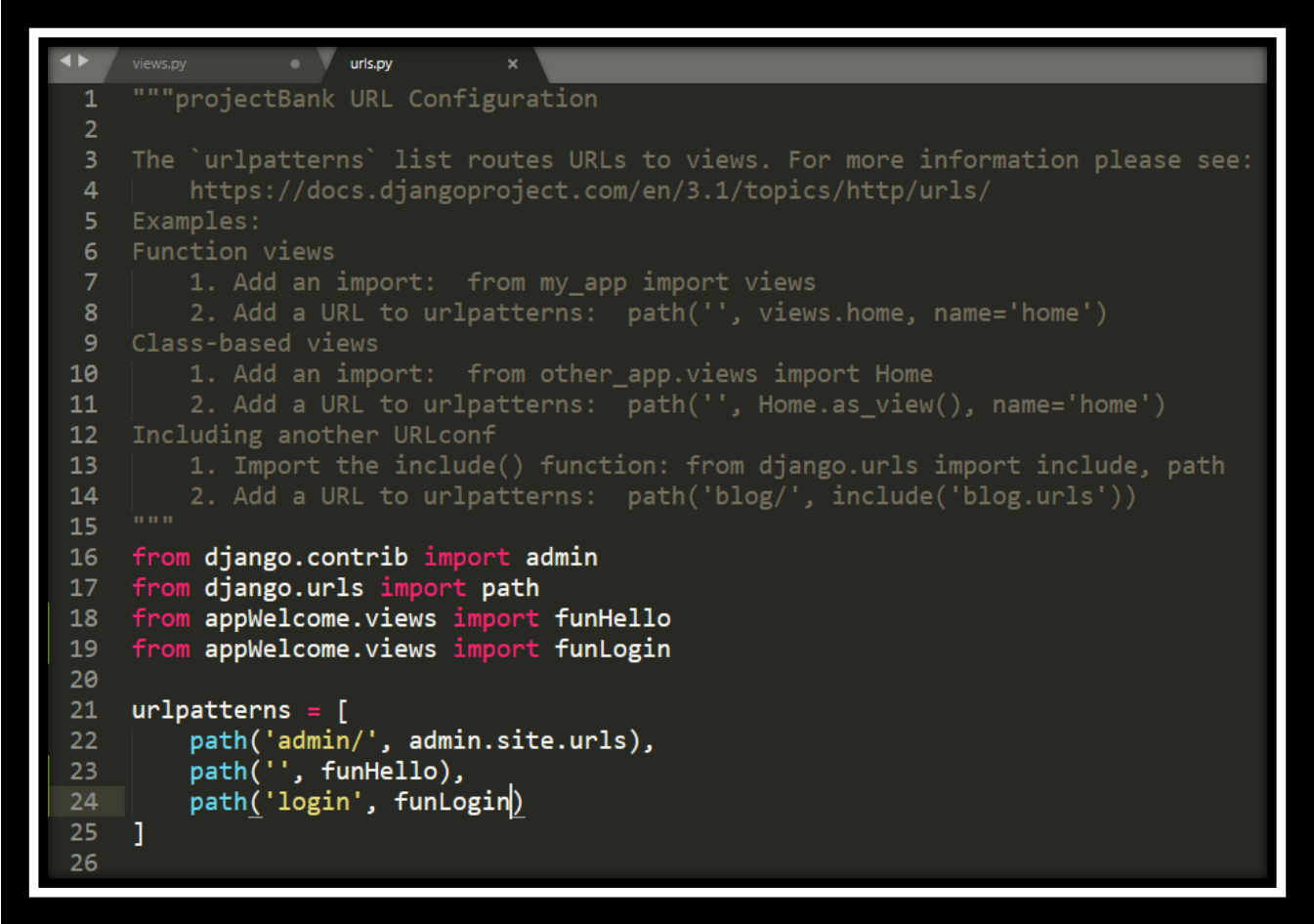
CREATING VIEWS

In order to create a user defined function we need to use view.py. For instance let us create a welcome view for the bank customers which can print a welcome message. In order to do this what we need to click on views.py and type the following code.

A screenshot of a code editor window with two tabs: 'views.py' and 'urls.py'. The 'views.py' tab is active, showing a Python file with the following code:

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 # Create your views here.
4 def funHello(request):
5     return HttpResponse("Hello Dear Customer, Welcome to SIB BANK!")
6 def funLogin(request):
7     return HttpResponse("<h1> Dear Customer, Pleae Login in to your account </h1>")
```

Now once you have created your functions lets go to `urls.py` which is present in your project (eg: here `projectDemo`) and create the path so that it can be accessed from the server.

A screenshot of a code editor with two tabs: 'views.py' and 'urls.py'. The 'urls.py' tab is active, showing Python code for URL configuration. The code includes a docstring with instructions on using 'urlpatterns', imports for 'admin', 'path', 'funHello', and 'funLogin', and a list of URL patterns for 'admin/', the root path, and 'login'.

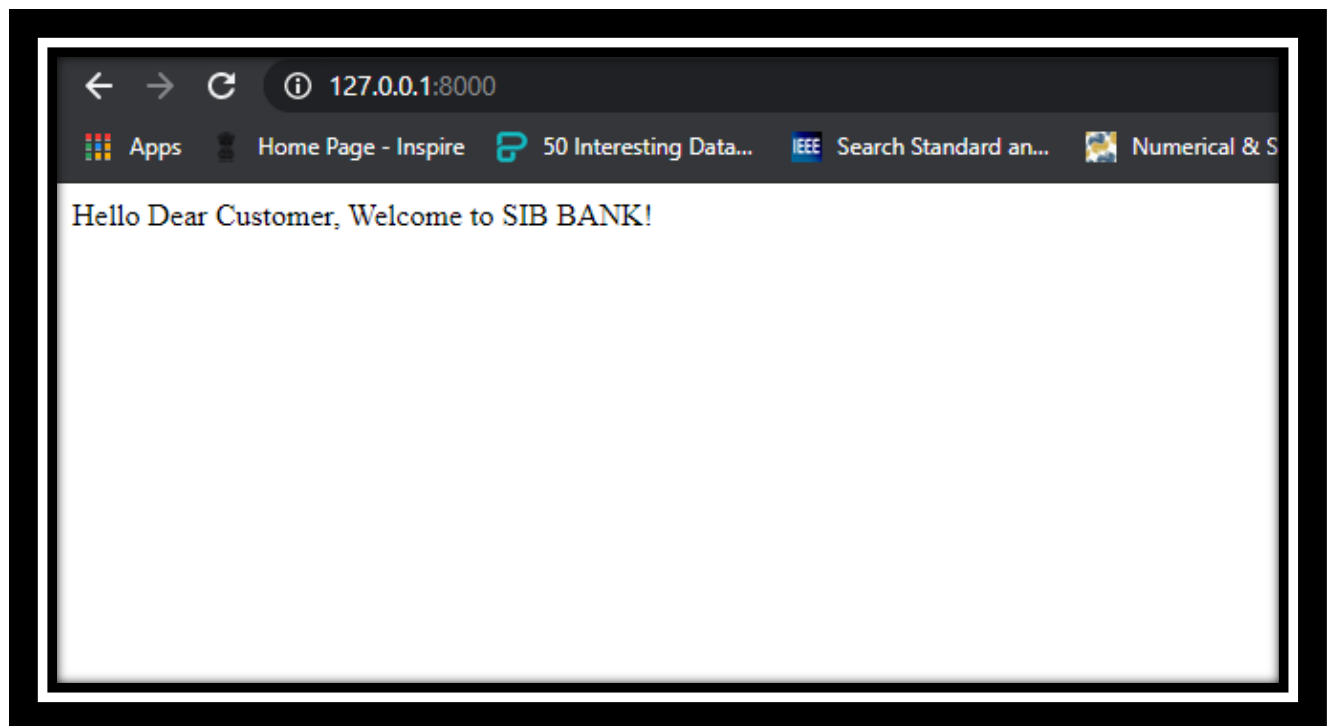
```
1 """projectBank URL Configuration
2
3 The `urlpatterns` list routes URLs to views. For more information please see:
4     https://docs.djangoproject.com/en/3.1/topics/http/urls/
5 Examples:
6 Function views
7     1. Add an import:  from my_app import views
8     2. Add a URL to urlpatterns:  path('', views.home, name='home')
9 Class-based views
10    1. Add an import:  from other_app.views import Home
11    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
12 Including another URLconf
13    1. Import the include() function: from django.urls import include, path
14    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path
18 from appWelcome.views import funHello
19 from appWelcome.views import funLogin
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', funHello),
24     path('login', funLogin)
25 ]
26
```

Once the code is written go to command prompt and run the server.

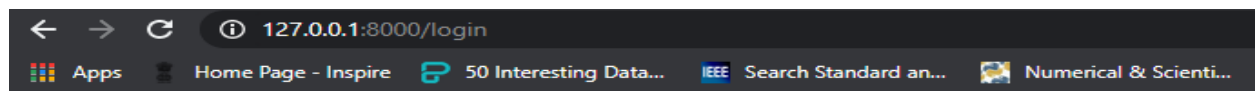
(MYENV) F:\00Django\projectDemo>python manage.py runserver

Done. Your web pages are visible now in your browser.

In the browser's address bar type <http://127.0.0.1:8000>



In a new tab of your web browser type <http://127.0.0.1:8000/login>



Dear Customer, Pleae Login in to your account

CREATING MODELS IN DJANGO

All the apps, models, pages and forms are to be created inside your environment. Every time when you open the cmd prompt and you are working with the existing project then the environment in which your project is residing should be activated by typing `<envname>/Scripts/activate`.

Step0: Go to the folder where the Django is installed, virtual environment is created.

>cd F:/00Django

Step1: Activate your virtual environment

Eg: MYENV/Scripts/activate

Step3: Go to app folder (Here appDemo folder)

In order to create a model go to you app folder which is inside your project folder. It is because model.py file exists only in app folder and the code related to it should be written there itself.

(MYENV) F:\00Django\projectDemo> cd appDemo

Step4: Write the code for creating models

What is a model?

A model is a visual representation of real world entities. In database terminology short it is collection of tables. Thus we need to write the code to create the tables. In Django model is always an object oriented model. Thus it is written using class concept.

Open models.py file present in your app (Here appDemo) folder and write a code for creating model

```

1 from django.db import models
2
3 # Create your models here.
4 class bankDetails(models.Model):
5     bankname = models.CharField(max_length=40)
6     address = models.CharField(max_length=40, blank=True, null=True)
7     DoE = models.DateTimeField() #Date of Establishment
8     bankID = models.IntegerField(null=False)
9     url = models.URLField(default='www.rbi.co.in')
10
11     def __str__(self):
12         return (self.bankname)
13
14 class bankAccount(models.Model):
15     username = models.CharField(max_length=40)
16     mybankname = models.ForeignKey(bankDetails, on_delete=models.CASCADE)
17     surname = models.CharField(max_length=40, blank=True, null=True)
18     emailid = models.EmailField(max_length=40)
19     DoI = models.DateTimeField() #Date of Initiation
20     accnumber = models.IntegerField()
21     url = models.URLField(default='www.google.com')
22
23     def __str__(self):
24         return (self.username+" account details")
25
26

```

In this code parent child tables are created.

Step5: Add the app name in settings.py file

The file called **settings.py** It holds all the configuration values that your web app needs to work; database settings, logging configuration, where to find static files, API keys if you work with external APIs, and a bunch of other stuff. .

After writing this code. Go to settings.py present in your project folder (Here projectDemo) in that go for the code related to 'INSTALLED_APPS' add your app name (Here the app name is appDemo).


```
views.py x settings.py models.py x urls.py admin.py x
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = '-hozlies^z)*gtx=616olz6%90)afrrp1b(*5v+$q=ia7ke#)e'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'appDemo' ### APP for which models are built
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49 ]
```

Step6: Import and register the model under admin.py file of your app

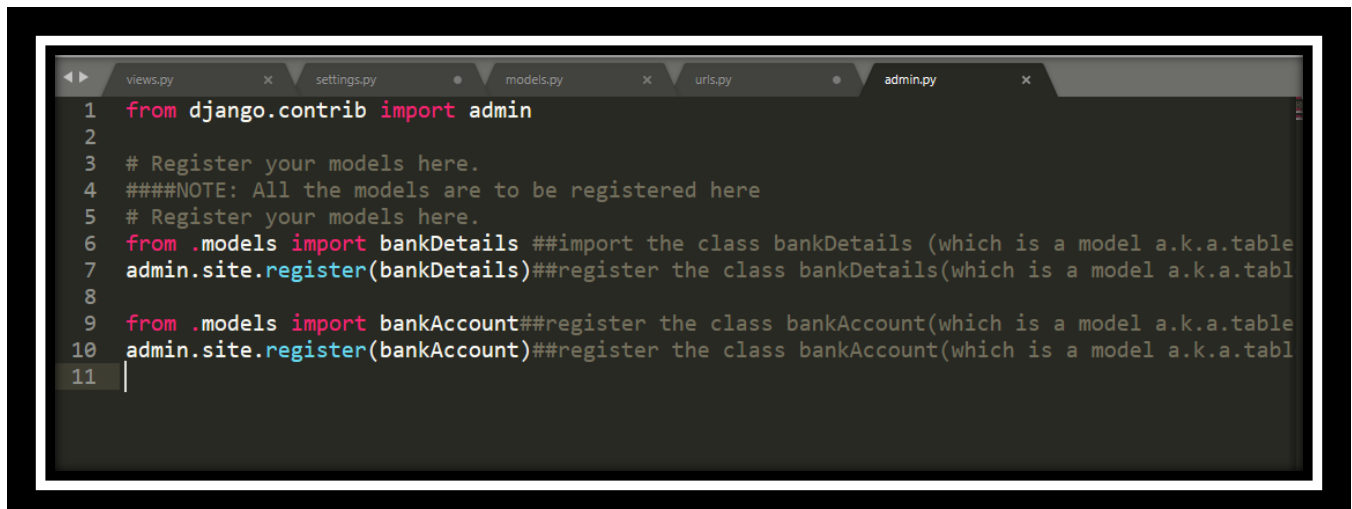
from . import model ### to avoid collision of import statements

from .models import bankDetails ###import the model

admin.site.register(bankDetails) ##register the model (class or aka table1)

from .models import bankAccount

admin.site.register(bankAccount) ##register the model (class or aka table1)



```
1 from django.contrib import admin
2
3 # Register your models here.
4 #####NOTE: All the models are to be registered here
5 # Register your models here.
6 from .models import bankDetails ##import the class bankDetails (which is a model a.k.a.table
7 admin.site.register(bankDetails)##register the class bankDetails(which is a model a.k.a.tabl
8
9 from .models import bankAccount##register the class bankAccount(which is a model a.k.a.table
10 admin.site.register(bankAccount)##register the class bankAccount(which is a model a.k.a.tabl
11
```

Step7: Save all the files

Step8: Go to cmd and type the following:

(MYENV) F:\00Django\projectDemo> python manage.py makemigrations

(MYENV) F:\00Django\projectDemo> python manage.py migrate

(MYENV) F:\00Django\projectDemo> python manage.py runserver

Step9: Start your admin page in web browser

Open any web browser and type 127.0.0.1:8000/admin

And refresh it. Here along with user authentication details you will find two tables (bankDetails and bankAccount) which you created by creating the models. Here you can add as many entries (rows) for bankDetails and bankAccount by entering the values for each attribute. You edit, delete and save aslo.

The same thing can be done using sqlite browser. Where you can alter the table, update or even delete the table.