Plural XT Test Merchant Application for Python-Django

**Install Python**

**Install Python: Windows**

Download Python for Windows from the website <https://www.python.org/downloads/windows/>.

Click on the "Latest Python 3 Release - Python x.x.x" link. If your computer is running a **64-bit** version of Windows, download the **Windows x86-64 executable installer**. Otherwise, download the **Windows x86 executable installer**. After downloading the installer, you should run it (double-click on it) and follow the instructions there.

One thing to watch out for: During the installation, you will notice a window marked "Setup". Make sure you tick the "**Add Python 3.7 to PATH**" checkbox and click on "Install Now".

Verify the installation was successful by opening a command prompt and running the python command:

command-line

$ **python --version**

Python 3.7.0

Note: On Windows, for python 3, instead of **python** in the above command, you can use **py** instead.eg:

$ **py** --version

Python 3.7.0

**Virtual environment**

Make a new directory say, **testdir**.

**mkdir testdir**

In the command prompt go to the location where your project(**testdir**) is present.

**cd testdir**

Make a virtual environment called **myvenv** :

**python -m venv myvenv**

Assuming you are still in directory **testdir**,start your virtual environment by running the following command:

command-line:

**myvenv\Scripts\activate**

You will know that you have virtualenv started when you see that the prompt in your console is prefixed with **(myvenv)**.

**Installing Django**

Now that you have your virtualenv started, you can install Django.

Before we do that, we should make sure we have the latest version of pip, the software that we use to install Django:

(Assuming that **testdir** is in D: drive)

command-line:

(myvenv) D:/testdir> **python -m pip install --upgrade pip**

Now, run the following command to install Django.

Command-line:

(myvenv) D:/testdir> **pip install django==3.1.1**

**Installing requests module**

Command-line:

(myvenv) D:/testdir> **pip install requests**

**Creating Django Project**

Run the following command. (Don't forget to add the period (or dot) **.** at the end):

Command-line:

(myvenv) D:/testdir> django-admin.exe startproject **mysite** **.**

**Changing settings**

Now, add the following lines, if not already present, at the end of the settings.py file of your project.

In file mysite/settings.py:

STATIC\_URL = '/static/'

STATIC\_ROOT = BASE\_DIR / 'static'

**Set up database**

Add the following lines, if not already present in file mysite/settings.py:

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

To create a database for our app, let's run the following in the console: (we need to be in the directory that contains the **manage.py** file which is **testdir** in this case)

(myvenv) D:/testdir> **python manage.py migrate**

**Including the**testmerchantapp **in your project**

Copy the testmerchantapp folder in **testdir** folder.

We also need to tell Django that it should use this app. We do that in the file mysite/settings.py by including it in the INSTALLED\_APPS.

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

**'testmerchantapp',** ]

**Create tables for models in your database**

First we have to make Django know that we have some changes in our model.

Go to your console window and type python manage.py makemigrations testmerchantapp. It will look like this:

(myvenv) D:/testdir> **python manage.py makemigrations testmerchantapp**

**Note:** Remember to save the files you edit. Otherwise, your computer will execute the previous version which might give you unexpected error messages

Django prepared a migration file for us that we now have to apply to our database.

Type the following in command-line:

(myvenv) D:/mysite> **python manage.py migrate testmerchantapp**

Output on command-line:

Operations to perform:

Apply all migrations: testmerchantapp

Running migrations:

Applying testmerchantapp.0001\_initial... OK

The models of the testmerchantapp are now in our database.

**Django URLs**

Edit the mysite/urls.py file to **add**the following lines(whichever are not present):

from django.urls import path**, include**

from django.contrib import admin

urlpatterns = [

path('admin/', admin.site.urls),

**path('testmerchant/', include('testmerchantapp.urls')),**

]

Assuming we run our server on [http://127.0.0.1:8000](http://127.0.0.1:8000/)/,

Django will now redirect everything that comes into '[http://127.0.0.1:8000/testmerchant](http://127.0.0.1:8000/)/' to testmerchantapp.urls and looks for further instructions there.

**Starting the web server**

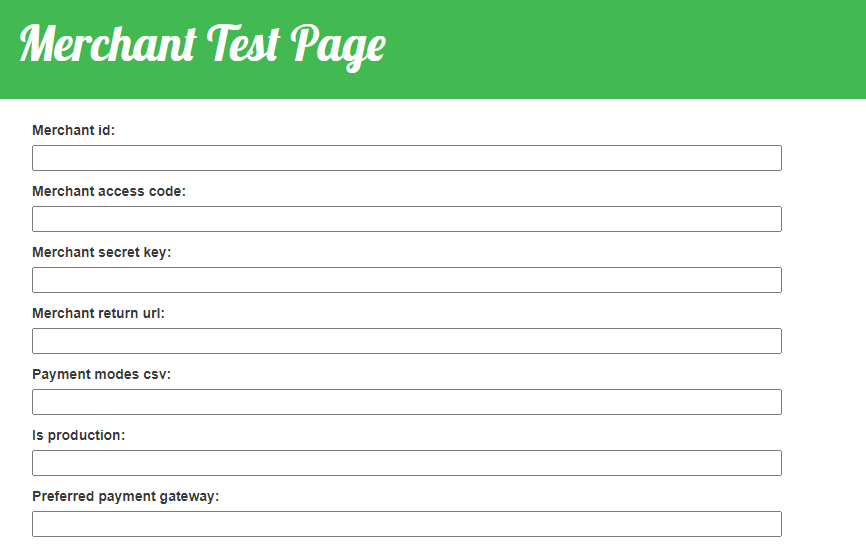
You need to be in the directory that contains the manage.py file. In the console, we can start the web server by running python manage.py runserver:

(myvenv) D:\testdir> **python manage.py runserver**

Now all you need to do is check that the website is running. Open your browser and enter this address:

**http://127.0.0.1:8000/testmerchant/MerchantTestPage/**

You will see a page similar to the following:



* + **Merchant id** – Set the Merchant Id provided by Pine Labs.
  + **Merchant access code** – Set the Merchant Access Code provided by Pine Labs.
  + Merchant secret key – Set the Merchant Secure Secret Key provided by Pine Labs.
  + **Merchant return url** – The url of the page where you want to handle the response from payment gateway. For this application, assuming the server runs at http://127.0.0.1:8000/, the value will be: ***http://127.0.0.1:8000/testmerchant/MerchantResponsePage/***
  + **Payment modes csv** – Enter the Payment Modes as Comma Separated Values. These Payment Modes will be shown on the Payment Gateway transaction page.

Some of the valid individual payment modes are:

CREDIT\_DEBIT

NETBANKING

EMI

UPI

WALLET

DEBIT\_EMI

ALL (For all payment modes)

So, for example, you can set: CREDIT\_DEBIT,WALLET,UPI

* + **Is production** – Set **1** for live/production operation and **0** for sandbox/test operation.
  + **Preferred payment gateway** – Set one of the following values as preferred payment gateway according to your preference:

(If you don’t have any preference, set it as **NONE**)

AMEX,

AMEX\_ENHANCED,

AXIS,

AXISB24,

BANKTEK,

BFL,

BHARATQR\_HDFC,

BILLDESK,

BOB,

CCAVENUE\_NET\_BANKING,

CITI,

CITRUS\_NET\_BANKING,

CORP,

DEBIT\_PIN\_FSS,

EBS\_NETBANKING,

EDGE,

FEDERAL,

FSS\_NETBANKING,

HDFC,

HDFC\_DEBIT\_EMI,

HDFC\_PRIZM,

HSBC,

ICICI,

ICICI\_SHAKTI,

IDBI,

LVB,

MASHREQ,

OPUS,

PAYTM,

PayU,

RAZOR\_PAY,

SBI,

SBI87,

SI\_HDFC,

SI\_PAYNIMO,

UBI,

UPI\_AXIS,

UPI\_HDFC,

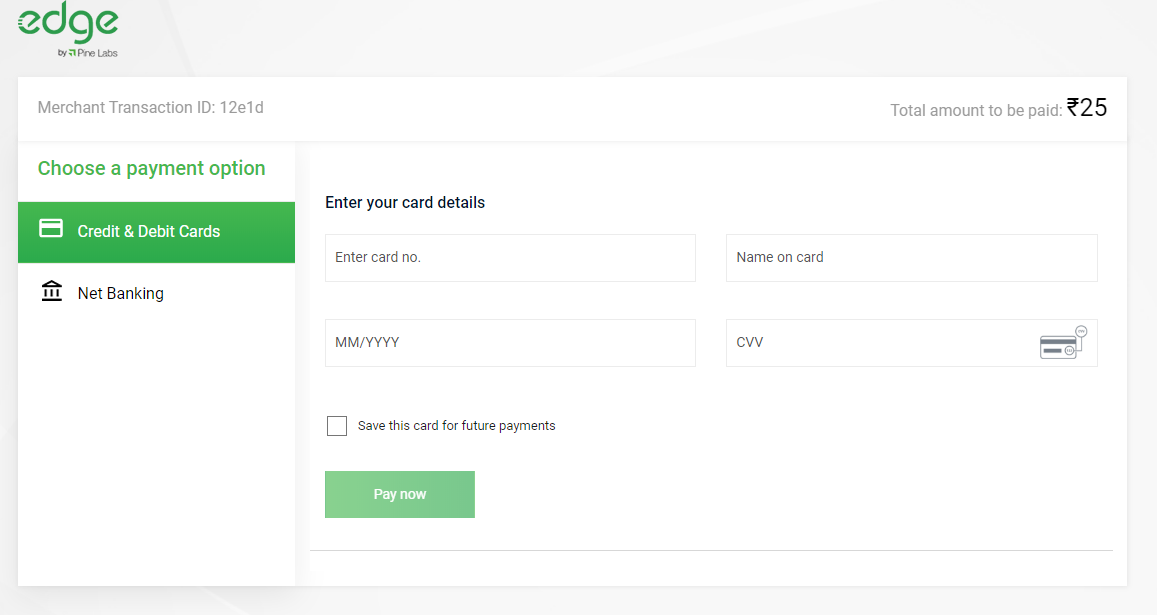
WALLET\_PAYZAPP,

WALLET\_PHONEPE,

YES,

ZEST\_MONEY

After filling the form and submitting the data, on successful order creation, you will be redirected to the payment gateway page:



After filling the payment details, the payment will be processed and you will be redirected to the Merchant return url page as specified in the form previously.

Back to the command prompt:

To **exit** from the virtualenv just type **deactivate**:

(myvenv) D:\mysite> **deactivate**