Python Expense Tracker Project

In this python django project, we will create an expense tracker that will take details of our expenses. While filling the signup form a person will also need to fill in the details about the income and the amount he/she wants to save. Some people earn on a daily basis, so their income can also be added on a regular basis. Details of expenses will be shown in the form of a pie chart on a weekly, monthly, and yearly basis. Installation of django is a must to start with the Expense Tracker project.

Project Prerequisites

Sound knowledge of django framework, html, css, javascript and python is required before starting this Expense Tracker project of Python.

Download Python Expense Tracker Project Code

Download source code of python expense tracker: **Expense Tracker Project Code**

Project File Structure

1. Install django framework
2. Create a project and an app
3. Models.py
4. Admin.py
5. Urls.py
6. Views.py

1. Install django framework:

To begin with the project, you need to install django on your system. To install django, write the following command on cmd or terminal window.

```
Pip install django
```

2. Create a project and an app:

We will create a new project named ExpenseTracker and an app to start the project. Write the following command on the terminal window.

```
django-admin startproject ExpenseTracker
python mange.py startapp home
```

Create a template and static folder to store your files. Template folder will contain all the html files. Static folder will contain all the css files ,images and javascript files.

3. Models.py

Database connectivity is done with the help of models.py. Create the following models in models.py file in the app of your project.

```
from django.db import models
from django.utils.timezone import now
from django.contrib.auth.models import User
from django.conf import settings
from django.db.models.signals import post_save
from django.dispatch import receiver
from django.db.models import Sum
#Create your models here.
SELECT_CATEGORY_CHOICES = [
("Food","Food"),
("Travel","Travel"),
("Shopping","Shopping"),
("Necessities","Necessities"),
("Entertainment","Entertainment"),
("Other","Other")
]
ADD_EXPENSE_CHOICES = [
("Expense","Expense"),
("Income","Income")
]
PROFESSION_CHOICES =[
("Employee","Employee"),
("Business","Business"),
("Student","Student"),
("Other","Other")
]
class Addmoney_info(models.Model):
user = models.ForeignKey(User,default = 1, on_delete=models.CASCADE)
add_money = models.CharField(max_length = 10 , choices = ADD_EXPENSE_CHOICES )
quantity = models.BigIntegerField()
Date = models.DateField(default = now)
Category = models.CharField( max_length = 20, choices = SELECT_CATEGORY_CHOICES , default ='Food')
class Meta:
db_table:'addmoney'
```

```
class UserProfile(models.Model):
user = models.OneToOneField(User,on_delete=models.CASCADE)
profession = models.CharField(max_length = 10, choices=PROFESSION_CHOICES)
Savings = models.IntegerField( null=True, blank=True)
income = models.BigIntegerField(null=True, blank=True)
image = models.ImageField(upload_to='profile_image',blank=True)
def __str__(self):
return self.user.username
```

Code Explanation:

SELECT_CATEGORY_CHOICES , EXPENSE_CHOICES , PROFESSION_CHOICES contain the list of options that will be given while filling the expense form.

a. Foreign key: It establishes many to one relationship.

b. Charfield():It stores small and large size strings in the database.

c. BigIntegerField():It can store numbers from -92