PROJECT DEVELOPMENT PHASE	SPRINT 4
TEAM ID	PNT2022TMID10365
PROJECT NAME	PERSONAL EXPENSE TRAKER APPLICATION

6. Views.py

a. Importing modules

```
from django.shortcuts import render,HttpResponse,redirect
from django.contrib import messages from
django.contrib.auth import authenticate ,logout from
django.contrib.auth import login as dj_login from
django.contrib.auth.models import User from .models
import Addmoney_info,UserProfile from
django.contrib.sessions.models import Session
from django.core.paginator import Paginator, EmptyPage , PageNotAnInteger
from django.db.models import Sum from django.http import JsonResponse
import datetime
from django.utils import timezone
```

Code Explanation:

a. Render: It returns the Httpresponse object and combines the template with the dictionary that is mentioned in it.
b. HttpResponse: It displays a text response to the user.
c. Redirect: It redirects the user to the specified url.
d. Messages: It helps to store and display messages to the user on the screen.
e. Authenticate: It verifies the user.
f. User: This model handles authentication as well as authorization.
g. Session: It helps the user to access only their data. Without sessions, every user's data will be displayed to the user. h. Paginator: It is used to manage paginated data.
   h. datetime:It is used to get the current date and time.

b. Login and Index function

```
def home(request):     if
request.session.has_key('is_logged'):
return redirect('/index')
   return render(request,'home/login.html')
# return HttpResponse('This is home') def
index(request):     if
request.session.has_key('is_logged'):
user_id = request.session["user_id"]
user = User.objects.get(id=user_id)
       addmoney_info = Addmoney_info.objects.filter(user=user).order_by('-Date')
paginator = Paginator(addmoney_info , 4)        page_number =
request.GET.get('page')
       page_obj = Paginator.get_page(paginator,page_number)
context = {
          # 'add_info' : addmoney_info,
         'page_obj' : page_obj
       }
   #if request.session.has_key('is_logged'):        return
render(request,'home/index.html',context)     return
redirect('home')
```

Code Explanation:

home() is a function that allows the user to access the dashboard once the user is logged in. index() function contains the backend of the dashboard page.

a. filter(): Queryset is filtered by filter().

b. get(): Single unique object can be obtained with get().

c. order_by(): It orders the queryset.

d. **OtherFunctios**

```
Def
      addmoney(reques
t):
  return render(request,'home/addmoney.html')
```

```
def profile(request):     if
request.session.has_key('is_logged'):
return render(request,'home/profile.html')
return redirect('/home')

def profile_edit(request,id):     if
request.session.has_key('is_logged'):
      add = User.objects.get(id=id)

      return render(request,'home/profile_edit.html',{'add':add})
return redirect("/home")
```

Code Explanation:
The first function redirects the user to the page where we can enter our expenses and income. profile() function redirects the user to the profile page where information of the user is displayed. profile_edit() redirects to the page where information of the user can be edited. These pages can only be accessed if the user is logged in.

d. Updating Profile

```
def profile_update(request,id):
   if request.session.has_key('is_logged'):        if request.method
== "POST":            user = User.objects.get(id=id)
user.first_name = request.POST["fname"]            user.last_name
= request.POST["lname"]            user.email =
request.POST["email"]            user.userprofile.Savings =
request.POST["Savings"]            user.userprofile.income =
request.POST["income"]            user.userprofile.profession =
request.POST["profession"]            user.userprofile.save()
user.save()            return redirect("/profile")
   return redirect("/home")
```

Code Explanation:
profile_update() function performs the backend of the edit profile form. User.objects.get() gets all the information of the user then all the updated information is saved again. This function is performed by save().

e. Signup, Login, and Logout backend:

```
def handleSignup(request):     if
request.method =='POST':            # get
the post parameters            uname =
request.POST["uname"]
fname=request.POST["fname"]
lname=request.POST["lname"]
email = request.POST["email"]
         profession = request.POST['profession']
Savings = request.POST['Savings']
income = request.POST['income']            pass1
= request.POST["pass1"]            pass2 =
request.POST["pass2"]
         profile = UserProfile(Savings = Savings,profession=profession,income=income)
         # check for errors in input
if request.method == 'POST':
try:
               user_exists = User.objects.get(username=request.POST['uname'])
messages.error(request," Username already taken, Try something else!!!")
return redirect("/register")                except User.DoesNotExist:                if
len(uname)>15:
                  messages.error(request," Username must be max 15 characters, Please try
again")
                  return redirect("/register")

               if not uname.isalnum():
                  messages.error(request," Username should only contain letters and
numbers, Please try again")
                  return redirect("/register")

               if pass1 != pass2:
                  messages.error(request," Password do not match, Please try again")
return redirect("/register")

         # create the user
         user = User.objects.create_user(uname, email, pass1)
user.first_name=fname            user.last_name=lname
user.email = email
         # profile = UserProfile.objects.all()

         user.save()
         # p1=profile.save(commit=False)
         profile.user = user
profile.save()
```

```
         messages.success(request," Your account has been successfully created")
return redirect("/")     else:
      return HttpResponse('404 - NOT FOUND ')
   return redirect('/login')

def handlelogin(request):     if request.method =='POST':         # get the post
parameters         loginuname = request.POST["loginuname"]
loginpassword1=request.POST["loginpassword1"]         user =
authenticate(username=loginuname, password=loginpassword1)         if
user is not None:
         dj_login(request, user)
request.session['is_logged'] = True             user
= request.user.id
request.session["user_id"] = user
         messages.success(request, " Successfully logged in")
return redirect('/index')         else:
         messages.error(request," Invalid Credentials, Please try again")
return redirect("/")
   return HttpResponse('404-not found') def
handleLogout(request):
      del request.session['is_logged']
      del request.session["user_id"]
logout(request)
      messages.success(request, " Successfully logged out")
return redirect('home')
```

Code Explanation:

handlesignup() function handles the backend of signup form. Uname, fname, lname, email , pass1, pass2, income, savings and profession will store the information of the form in these variables.

Various conditions are there to sign up . The username should be unique, pass1 and pass 2 should be the same and also the length of the username should be maximum 15 characters. handlelogin() handles the backend of the login page. If the information entered by the user is correct, the user will be redirected to the dashboard. handleLogout() handles the backend of logout.

a. error(): This function gives the error message on the screen if a condition is not satisfied.

b. len():This function returns the length of the string, array, dictionary etc.

c. success():If a condition is satisfied, it displays the message that is specified in the parentheses.

f. Add Money Form and Add Money Update Backend:

```
def addmoney_submission(request):     if
request.session.has_key('is_logged'):
if request.method == "POST":
          user_id = request.session["user_id"]
user1 = User.objects.get(id=user_id)
addmoney_info1 =
Addmoney_info.objects.filter(user=user1).order_by('-Date')
add_money = request.POST["add_money"]           quantity
= request.POST["quantity"]           Date =
request.POST["Date"]           Category =
request.POST["Category"]
          add = Addmoney_info(user =
user1,add_money=add_money,quantity=quantity,Date = Date,Category=
Category)
          add.save()
          paginator = Paginator(addmoney_info1, 4)
page_number = request.GET.get('page')
          page_obj = Paginator.get_page(paginator,page_number)
context = {
            'page_obj' : page_obj
            }
          return render(request,'home/index.html',context)
return redirect('/index') def
addmoney_update(request,id):     if
request.session.has_key('is_logged'):        if
request.method == "POST":
          add          =     Addmoney_info.objects.get(id=id)
add     .add_money     =     request.POST["add_money"]
```

```
add.quantity = request.POST["quantity"]          add.Date
=  request.POST["Date"]                          add.Category  =
request.POST["Category"]            add .save()
          return redirect("/index")
return redirect("/home")
```

Code Explanation:

addmoney_submission() handles the backend of the form we filled for our daily expenses. addmoney_update() saves the information of the form after we have edited .

g. Expense Edit and Expense Delete Backend:

```
def expense_edit(request,id):     if
request.session.has_key('is_logged'):
addmoney_info = Addmoney_info.objects.get(id=id)
      user_id      =      request.session["user_id"]
user1 = User.objects.get(id=user_id)      return
render(request,'home/expense_edit.html',{'addmoney_info':addmoney_info})
return redirect("/home")

def expense_delete(request,id):     if
request.session.has_key('is_logged'):
addmoney_info = Addmoney_info.objects.get(id=id)
      addmoney_info.delete()
return redirect("/index")
   return redirect("/home")
```

Code Explanation:

expense_edit() form redirects the user to the edit form and also extracts the details of the user from the database and displays it on the screen. expense_delete() helps in deleting the expenses.

h. Monthly, weekly , yearly expense Backend

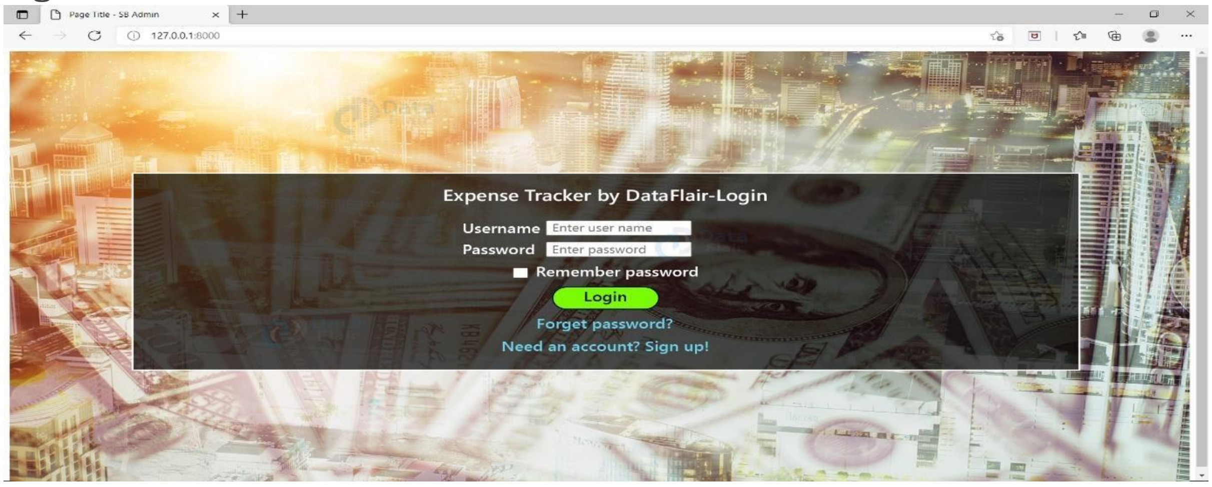
```
def expense_month(request):
   todays_date = datetime.date.today()
   one_month_ago = todays_date-datetime.timedelta(days=30)
   user_id = request.session["user_id"]     user1 =
User.objects.get(id=user_id)     addmoney =
Addmoney_info.objects.filter(user =
user1,Date__gte=one_month_ago,Date__lte=todays_date)
finalrep ={}

   def get_Category(addmoney_info):
      # if addmoney_info.add_money=="Expense":
      return addmoney_info.Category
   Category_list = list(set(map(get_Category,addmoney)))

   def get_expense_category_amount(Category,add_money):
quantity = 0
      filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
for item in filtered_by_category:
         quantity+=item.quantity
return quantity

   for x in addmoney:
for y in Category_list:
         finalrep[y]= get_expense_category_amount(y,"Expense")

   return JsonResponse({'expense_category_data': finalrep}, safe=False)


def stats(request):     if
request.session.has_key('is_logged') :
todays_date = datetime.date.today()
one_month_ago = todays_date-
datetime.timedelta(days=30)
      user_id = request.session["user_id"]        user1 =
User.objects.get(id=user_id)        addmoney_info =
Addmoney_info.objects.filter(user =
user1,Date__gte=one_month_ago,Date__lte=todays_date)
sum = 0         for i in addmoney_info:           if
i.add_money == 'Expense':
            sum=sum+i.quantity
      addmoney_info.sum = sum        sum1 = 0         for i in addmoney_info:
if i.add_money == 'Income':              sum1 =sum1+i.quantity
addmoney_info.sum1 = sum1        x=
user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum        y=
user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum        if
x<0:
         messages.warning(request,'Your expenses exceeded your savings')
x = 0        if x>0:           y = 0
      addmoney_info.x = abs(x)
addmoney_info.y = abs(y)
```

```
        return render(request,'home/stats.html',{'addmoney':addmoney_info})

def expense_week(request):
    todays_date = datetime.date.today()
    one_week_ago = todays_date-datetime.timedelta(days=7)
    user_id = request.session["user_id"]     user1 =
User.objects.get(id=user_id)     addmoney =
Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)
finalrep ={}

    def get_Category(addmoney_info):
return addmoney_info.Category
    Category_list = list(set(map(get_Category,addmoney)))


    def get_expense_category_amount(Category,add_money):
quantity = 0
        filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
for item in filtered_by_category:
            quantity+=item.quantity
return quantity

    for x in addmoney:
for y in Category_list:
finalrep[y]=
get_expense_category_a
mount(y,"Expense")

    return JsonResponse({'expense_category_data': finalrep}, safe=False)

def weekly(request):     if
request.session.has_key('is_logged') :
todays_date = datetime.date.today()
        one_week_ago = todays_date-datetime.timedelta(days=7)
        user_id = request.session["user_id"]        user1 =
User.objects.get(id=user_id)        addmoney_info =
Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)
sum = 0         for i in addmoney_info:           if
i.add_money == 'Expense':
              sum=sum+i.quantity
        addmoney_info.sum = sum        sum1 = 0         for i in addmoney_info:
if i.add_money == 'Income':               sum1 =sum1+i.quantity
addmoney_info.sum1 = sum1        x=
user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum        y=
user1.userprofile.Savings+addmoney_info.sum1 - addmoney_info.sum        if
x<0:
          messages.warning(request,'Your expenses exceeded your savings')
x = 0        if x>0:            y = 0
       addmoney_info.x = abs(x)
addmoney_info.y = abs(y)
   return render(request,'home/weekly.html',{'addmoney_info':addmoney_info})

def check(request):     if request.method == 'POST':
user_exists = User.objects.filter(email=request.POST['email'])
messages.error(request,"Email not registered, TRY AGAIN!!!")
return redirect("/reset_password")

def info_year(request):     todays_date
= datetime.date.today()
   one_week_ago = todays_date-datetime.timedelta(days=30*12)
   user_id = request.session["user_id"]     user1 =
User.objects.get(id=user_id)     addmoney =
Addmoney_info.objects.filter(user =
user1,Date__gte=one_week_ago,Date__lte=todays_date)
finalrep ={}

   def get_Category(addmoney_info):
       return addmoney_info.Category
   Category_list = list(set(map(get_Category,addmoney)))


   def get_expense_category_amount(Category,add_money):
quantity = 0
       filtered_by_category = addmoney.filter(Category = Category,add_money="Expense")
for item in filtered_by_category:
          quantity+=item.quantity
return quantity

   for x in addmoney:
for y in Category_list:
          finalrep[y]= get_expense_category_amount(y,"Expense")

   return JsonResponse({'expense_category_data': finalrep}, safe=False)

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

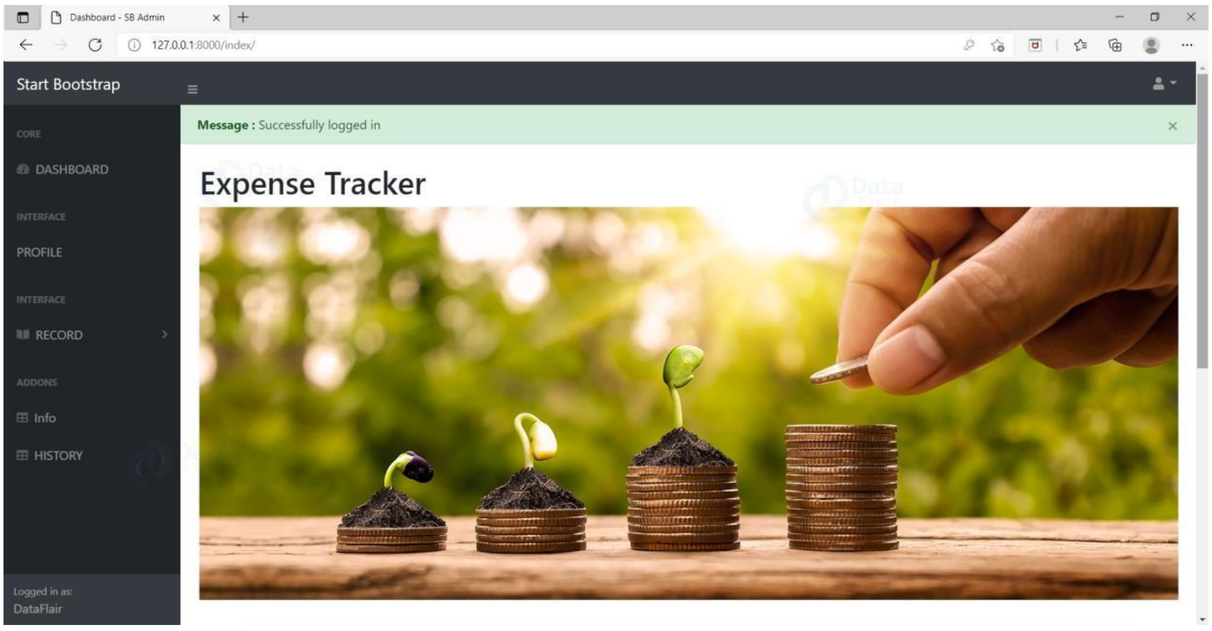
Code Explanation:
expense_month() function gets the data of the expenses of the current month. get_category() function gets the category (expense/income) from the database. get_expense_category_amount() fetches the amount from the database of the category(expense). stats() function calculates the overall expenses and savings made by the user in a month. expense_week() and info_year() performs the same function as expense_month() but on a weekly basis. weekly() gets the amount saved in a month and also the overall expenses of a user.

Python Expense Tracker Output:

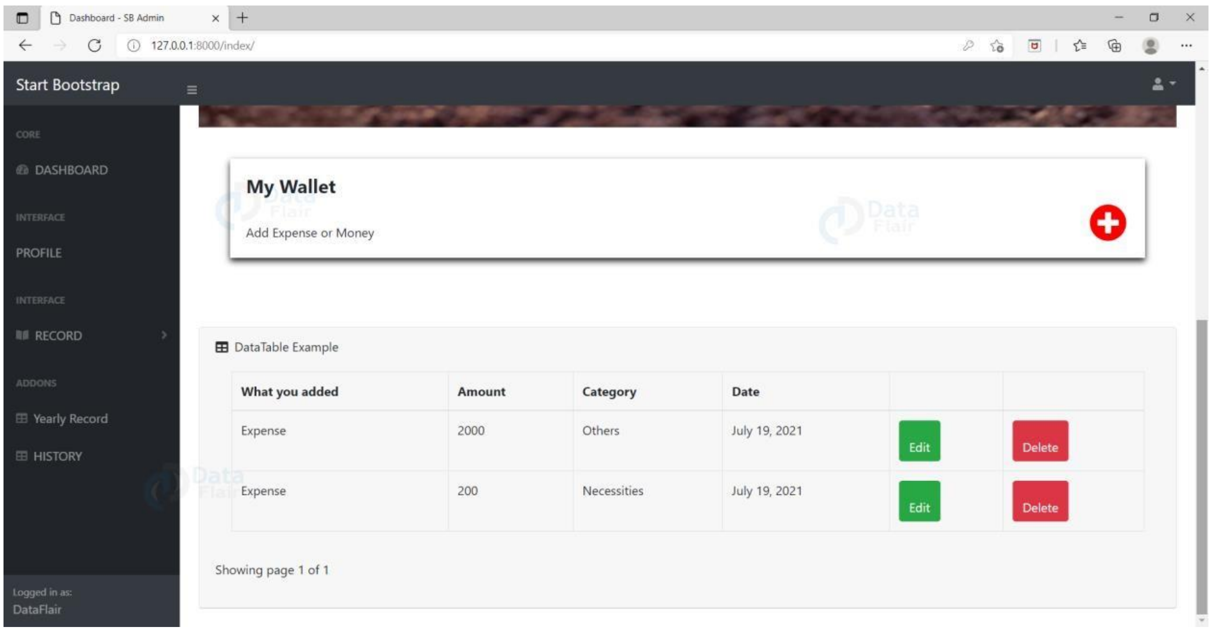
Login Form:



Dashboard:



Message : Successfully logged in



My Wallet

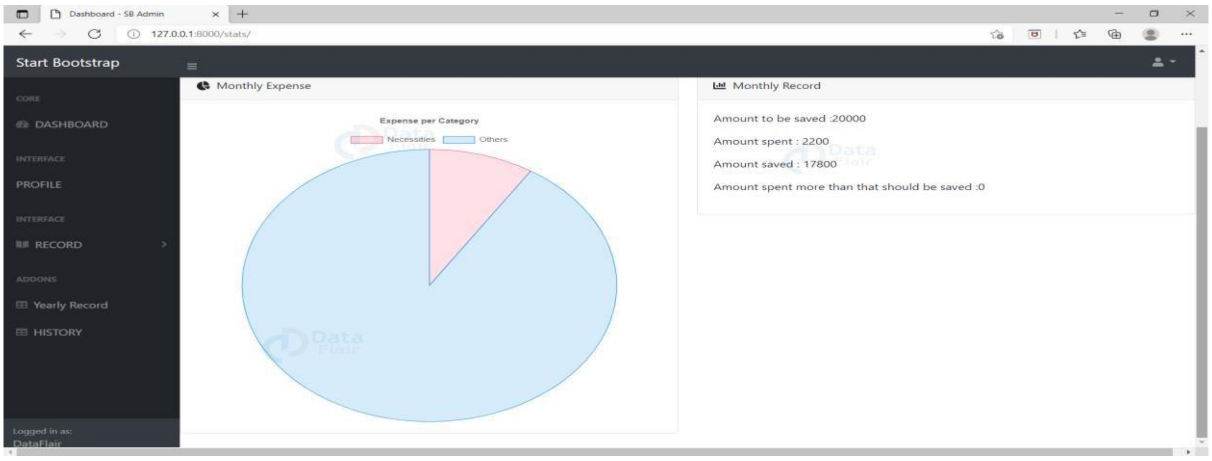
Add Expense or Money

DataTable Example

What you added	Amount	Category	Date		
Expense	2000	Others	July 19, 2021	Edit	Delete
Expense	200	Necessities	July 19, 2021	Edit	Delete

Showing page 1 of 1

Monthly Expense Page:



Monthly Record

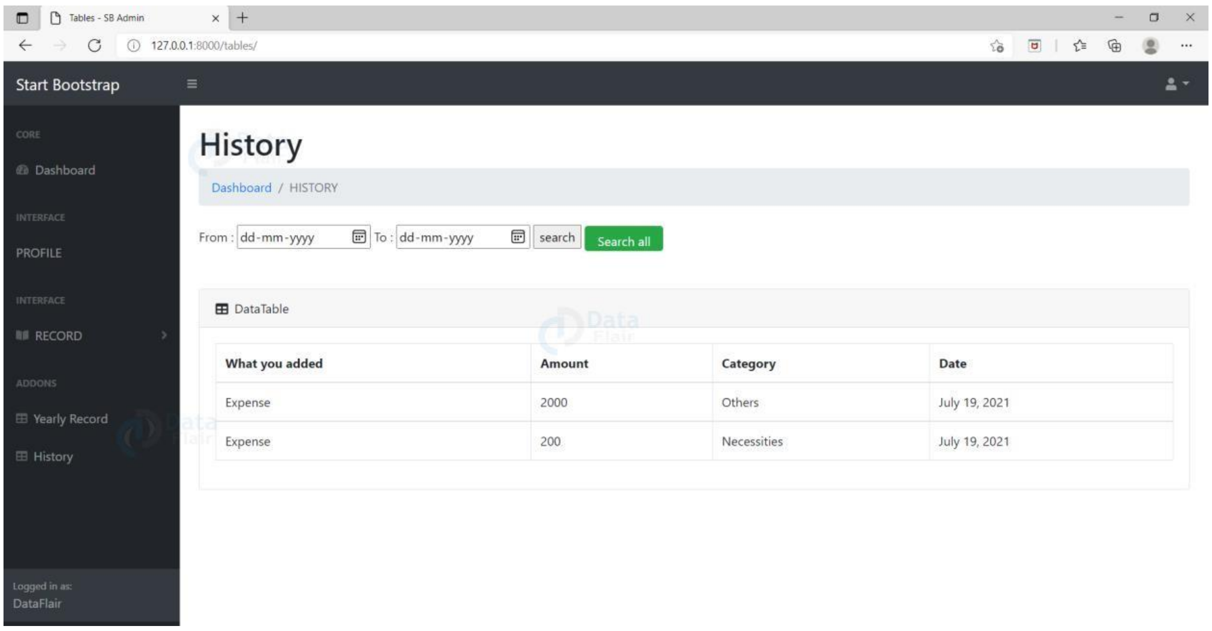
Amount to be saved :20000

Amount spent : 2200

Amount saved : 17800

Amount spent more than that should be saved :0

History Page:



History

Dashboard / HISTORY

What you added	Amount	Category	Date
Expense	2000	Others	July 19, 2021
Expense	200	Necessities	July 19, 2021

Summary

We have successfully created the expense tracker project in python. We learned a variety of concepts while making this project.