

📈 Financial_Api_Project Report

prepared by:

-Name: [Asala Shawky]

-Date: [4/22/2025]

Project Overview:

Managing financial transactions using

Python

Django

Django Rest Framework (DRF)

Testing on

Postman

Implemented Endpoints:

Endpoint

Methods

Description

| `server url` /api/transactions/` | GET, POST | عرض وإنشاء معاملات مالية

| `server url` /api/transactions/<id>/` | GET, PUT, DELETE | ID عرض، تعديل، أو حذف معاملة حسب الـ

|

| `server url` /api/users/` | GET | عرض جميع المستخدمين

| `server url` /api/users/<user_id>/transactions/` | GET | عرض المعاملات الخاصة بمستخدم معين

| `server url` /api/reports/?user_id=&year=&month=` | GET | تقرير إجمالي المعاملات لمستخدم معين **بالسنة أو الشهر**

 Testing Strategy:

 Tools Used:

- **Postman**: لتنفيذ اختبار كل الـ endpoints.

 Test Cases Covered:

- إنشاء معاملة مالية (POST)

- عرض جميع المعاملات (GET)

- عرض معاملة واحدة (GET by ID)

- تعديل معاملة (PUT)

- حذف معاملة (DELETE)

- عرض المستخدمين (GET)

- عرض معاملات مستخدم (GET)

- استخراج تقرير سنوي/شهري (GET with params)

 Deliverables:

1. **Source code** (` .zip ` أو GitHub repo).

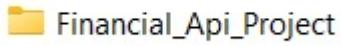
2. **Postman Collection** (` .json ` export).

3. **Development Report** (.pdf).

Now, I will show you step-by-step

How to create a comprehensive financial API

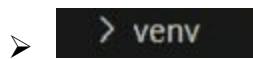
1. Create a folder of the project



2. Open folder in <vsc< go to terminal< make virtual environment

- Pip install virtualenv

- python -m virtualenv venv



3. Activate the virtual environment

- venv\ scripts\activate

- if didn't work use : **set-ExecutionPolicy RemoteSigned –scope CurrentUser**

4. download frameworks

- pip install Django djangorestframework

```

PS D:\asala\finance_API> Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
PS D:\asala\finance_API> venv\Scripts\activate
(venv) PS D:\asala\finance_API> pip install django djangorestframework
Collecting django
  Downloading Django-5.2-py3-none-any.whl (8.3 MB)
    8.3/8.3 MB 1.9 MB/s eta 0:00:00Collecting djangorestframework
  Downloading djangorestframework-3.16.0-py3-none-any.whl (1.1 MB)
    1.1/1.1 MB 1.6 MB/s eta 0:00:00Collecting asgiref>=3.8.1
  Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Collecting sqlparse>=0.3.1
  Downloading sqlparse-0.5.3-py3-none-any.whl (44 kB)
    44.4/44.4 kB 2.3 MB/s eta 0:00:00Collecting tzdata
  Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
    347.8/347.8 kB 1.8 MB/s eta 0:00:00Installing collected packages: tzdata, sqlparse, asgiref, django, djangorestframework
Successfully installed asgiref-3.8.1 django-5.2 djangorestframework-3.16.0 sqlparse-0.5.3 tzdata-2025.2

[notice] A new release of pip available: 22.3.1 → 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

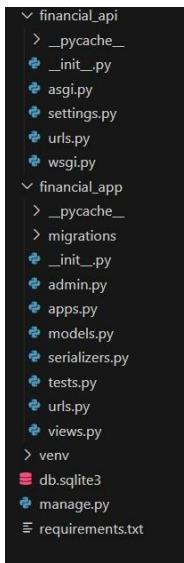
```

5. pip freeze > requirements.txt

6. make apps, Api, migration

- Django-admin startproject financial_app
- Django-admin startapp financial_project
- Python manage.py makemigrations
- Python manage.py migrate

7. apps will be appear here :



8. create superuser (admin)

- python manage.py createsuperuser
- put : username,email,password

9. go to < financial.api < settings.py < INSTALLED_APPS

- put : 'rest_framework', 'financial_app'

10. https://github.com/asala810/financial_api_project.git

11. Run server:

- Python manage.py runserver
- Apply live server

12. Time to test :

- Put << <http://127.0.0.1:8000/api/transactions/>
- Put << <http://127.0.0.1:8000/admin/>
- Put << <http://127.0.0.1:8000/api/users/>



13.



14.

The screenshot shows a Django REST framework API endpoint titled "Transaction List Create". At the top, there is a header bar with "Django REST framework" and a "Transaction List Create" title. Below the header, there is a "GET /api/transactions/" section showing a response with status "HTTP 200 OK" and content type "application/json". The response body contains a JSON array with one item, representing a transaction. The transaction has an ID of 2, a transaction date of "2025-04-21T20:27:37.854766Z", a description of "Test Transaction", an amount of "100.00", and a user ID of 1. Below this, there is a "Raw data" and "HTML form" tab. The "HTML form" tab displays three input fields: "Description", "Amount", and "User id".

15.

16. Time to test on postman

The screenshot shows a Postman workspace with a collection named "Overview". A POST request is being made to "http://127.0.0.1:8000/api/transactions/". The request body is set to "JSON" and contains the following JSON data:

```

1 {
2   "description": "Test Deposit",
3   "amount": 500.00,
4   "user_id": 1
5 }

```

The response is a 201 Created status with a response body containing the created transaction details:

```

1 {
2   "id": 5,
3   "transaction_date": "2025-04-22T10:26:34.862688Z",
4   "description": "Test Deposit",
5   "amount": "500.00",
6   "user_id": 1
7 }

```

Post. Request

- Headers >> Content-Type: application/json

➤ **Body (raw + JSON):**

```
{  
  "description": "Test Deposit",  
  "amount": 500.00,  
  "user_id": 1  
}
```

The screenshot shows the Postman application interface. At the top, there's a header bar with 'Workspaces', 'More', a search icon, and an 'Upgrade' button. Below the header, a collection named 'Overview' is selected, with the URL 'http://127.0.0.1:8000/api/' listed. A 'Send' button is visible on the right. The main area shows a 'GET' request to 'http://127.0.0.1:8000/api/transactions/'. The response status is '200 OK' with a duration of '9 ms' and a size of '811 B'. The response body is displayed as JSON, showing five transaction records. The JSON output is as follows:

```
3 |   {  
4 |     "id": 2,  
5 |     "transaction_date": "2025-04-21T20:27:37.854766Z",  
6 |     "description": "Test Transaction",  
7 |     "amount": "100.50",  
8 |     "user_id": 1  
9 |   },  
10 |   {  
11 |     "id": 3,  
12 |     "transaction_date": "2025-04-22T10:14:44.617096Z",  
13 |     "description": "Test Transaction",  
14 |     "amount": "100.50",  
15 |     "user_id": 1  
16 |   },  
17 |   {  
18 |     "id": 4,  
19 |     "transaction_date": "2025-04-22T10:17:56.300165Z",  
20 |     "description": "Test Transaction",  
21 |     "amount": "100.50",  
22 |     "user_id": 1  
23 |   },  
24 |   {  
25 |     "id": 5,  
26 |     "transaction_date": "2025-04-22T10:26:34.862688Z",  
27 |     "description": "Test Deposit",  
28 |     "amount": "500.00",  
      "user_id": 1  
}
```

Get. Request

The screenshot shows two separate Postman sessions side-by-side.

Top Session (POST Request):

- Method:** POST
- URL:** http://127.0.0.1:8000/api/transactions/
- Body (JSON):**

```
1 {
2     "id": 6,
3     "transaction_date": "2025-04-22T10:30:45.810754Z",
4     "description": "second Deposit",
5     "amount": "500.00",
6     "user_id": 1
7 }
```

Bottom Session (GET Request):

- Method:** GET
- URL:** http://127.0.0.1:8000/api/transactions/2
- Headers (7):**

Key	Value	Description
Key	Value	Description

- Body (JSON):**

```
1 {
2     "id": 2,
3     "transaction_date": "2025-04-21T20:27:37.854766Z",
4     "description": "Test Transaction",
5     "amount": "100.50",
6     "user_id": 1
7 }
```

To get specific data about user(1or2or3or,etc)

Overview | GET http://127.0.0.1:8000/api/users/ | No environment

New Collection / http://127.0.0.1:8000/api/transactions/ post request | Save | Share

GET http://127.0.0.1:8000/api/users/ Send

Params Auth Headers (7) Body Scripts Settings Cookies

Headers 7 hidden

Key	Value	D...	Bulk Edit	Presets
Key	Value	Description		

Body 200 OK 4 ms 392 B Save Response

{ } JSON ▾ Preview Visualize

```

1 [
2   {
3     "id": 1,
4     "username": "asala1",
5     "email": "asalashawkyhana810@gmail.com"
6   }
7 ]

```

Overview | GET http://127.0.0.1:8000/api/users/1/transactions/ | No environment

New Collection / http://127.0.0.1:8000/api/transactions/ get request | Save | Share

GET http://127.0.0.1:8000/api/users/1/transactions/ Send

Params Auth Headers (7) Body Scripts Settings Cookies

Body 200 OK 7 ms 924 B Save Response

{ } JSON ▾ Preview Visualize

```

9 [
10   {
11     "id": 3,
12     "transaction_date": "2025-04-22T10:14:44.617096Z",
13     "description": "Test Transaction",
14     "amount": "100.50",
15     "user_id": 1
16   },
17   {
18     "id": 4,
19     "transaction_date": "2025-04-22T10:17:56.300165Z",
20     "description": "Test Transaction",
21     "amount": "100.50",
22     "user_id": 1
23   },
24   {
25     "id": 5,
26     "transaction_date": "2025-04-22T10:26:34.862688Z",
27     "description": "Test Deposit",
28     "amount": "500.00",
29     "user_id": 1
30   },
31   {
32     "id": 6,
33     "transaction_date": "2025-04-22T10:30:45.810754Z",
34     "description": "second Deposit",
35     "amount": "500.00",
36     "user_id": 1
37   }
38 ]

```

Every data we post , stored in the ‘get request’

HTTP ... / http://127.0.0.1:8000/api/users/1/transactions/ get request

GET http://127.0.0.1:8000/api/reports/?user_id=1&year=2025&month=4

Send

Params Auth Headers (7) Body Scripts Settings Cookies

200 OK 7 ms 408 B Save Response

```
{
  "user_id": "1",
  "year": "2025",
  "month": "4",
  "total_amount": 1301.5,
  "transaction_count": 5
}
```

To report all transactions of specific user along the year, month ,

Report total amount

HTTP ... / http://127.0.0.1:8000/api/reports/?user_id=1&year=2025&month=4

PUT http://127.0.0.1:8000/api/transactions/2/

Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

```
{
  "description": "Updated Payment",
  "amount": 999.99,
  "user_id": 2
}
```

Body Save Response

200 OK 47 ms 463 B Save Response

```
{
  "id": 2,
  "transaction_date": "2025-04-21T20:27:37.854766Z",
  "description": "Updated Payment",
  "amount": 999.99,
  "user_id": 2
}
```

If you want to update data of specific user, with his id

The screenshot shows the Postman application interface. At the top, there's a header with 'Overview', 'New...', 'DEL http://127.0.0.1:8000/api/transactions/2 put request', 'Save', and 'Share' buttons. Below the header, a search bar has 'DELETE' selected and the URL 'http://127.0.0.1:8000/api/transactions/6/'. A 'Send' button is to the right. Underneath, tabs for 'Params', 'Auth', 'Headers (7)', 'Body', 'Scripts', 'Settings', and 'Cookies' are visible, with 'Headers' currently selected. A 'Query Params' table is present but empty. In the main body area, there's a 'Body' tab with '204 No Content' status, '46 ms' response time, and '318 B' size. Below this, there are buttons for 'JSON', 'Preview', 'Visualize', and other tools. The response content area shows the number '1'. On the far left, there's a small navigation icon.

Delete data of specific user

Now you have all details

Challenges that faced me : how to upload folders on GitHub

And make a link between GitHub and visual studio code