FISCHERJORDAN

Backend Developer (Django/Python)
Round 2 — Take Home Assignment

Personal Finance Tracker

Instructions

Over the next 5 days, you'll be building a *'Personal Finance Tracker'* — a comprehensive tool to manage and analyze a user's financial health. Before you get started, a couple of instructions:

1. The take-home assignment has been broken into two parts — (a) the basic task, and (b) extra credit. For evaluation purposes, it is **only** compulsory to attempt all sub-parts of (a) the basic task. The nature of extra credit is such that you only get points for extra credit **if and only if** all sub-parts of (a) the basic task have been completed and are fully functional.

2. You must **exclusively** use Django (Django REST Framework, Django Templating Language, Python) to build and complete this assignment.

3. You have 5 days to complete this assignment. You **are** allowed to use any internet resources *(eg. ChatGPT, StackOverflow, etc).*

4. Create a **private** repository on Github titled *'FJ-BE-R2 <Your-Name> <Your-College>'* and use this repository over the duration of this take-home assignment. Share this repository with @psrth and @paramkpr.

5. Evaluation is extremely subjective and is done on an individual basis. However, the general parameters that we look for are (in no particular order): percentage of the problem attempted, functionality, code readability, documentation, logic, code efficiency, user experience, extra initiative.

6. That's it — let's get started!

Part A — The Basic Task

Your task is to develop a web application where users can track their income, expenses, and investments. Users should be able to get insights and generate reports on their financial standing.

The development of this project will be phased to ensure structured progress.

Day 1-2: Basic Functionality

1. User Authentication: Use Django's in-built authentication system to implement user authentication. The users should be able to register, log in, and manage their profiles.

2. Database Structure: Create a well-defined database structure and implement it using Django models. The database should support the tracking of various financial details, including:
    a. Income sources and the amount from each source
    b. Expense categories and the amount spent in each
    c. An item (or transaction) should have a date, amount, and a description

3. Transaction Management: Allow users to add, edit, and delete income and expense transactions.

4. Dashboard: Develop a user-friendly dashboard that provides an overview of the user's financial status, including graphical representations of income, expenses, and savings.

5. Reporting: Users should be able to generate reports on their financial data, such as monthly income vs. expenses report.

6. Budgeting: Allow users to set budget goals for different expense categories and track their progress.

Day 3: Additional Features

1. OAuth Integration: Allow users to sign up using Google through Django Allauth or Google's API client library.

2. Notification System: Set up a system to notify users about budget overruns, etc., using email notifications through Sendgrid or another service.

3. Database Upgrade: Transition from SQLite to MySQL to enhance the robustness of your database setup.
4. Expense Splitting: Introduce a feature to split expenses with others and keep track of who owes whom.
5. Receipt Uploading: Allow users to upload and store receipts for their transactions.

Day 4: Deployment

Deploy your Django application to a production server to make it accessible online. You might consider platforms like DigitalOcean, AWS, or Heroku for deployment. Ensure the security and performance optimization of the application during deployment.

Day 5: Testing

Make sure that what you've built works reliably!

Part B — Additional Features

- Recurring Transactions: Implement functionality to add recurring transactions, such as monthly bills.
- Currency Conversion: Allow users to manage transactions in different currencies and convert them to a base currency for reporting.
- Tax Calculation: Help users calculate their taxes based on their financial data.
- Integration with Banking APIs: If possible, integrate with banking APIs to fetch transaction data automatically.
- Integration with OpenAI: Any LLM-based integration that you can think of.

Submission

Google Form https://forms.gle/uKfPQUG9BR1QkbbQ8, Submit the assignment here:

1. Link to the deployed, working demo.
2. Link to your private Github repository.
3. List of tasks you were able to complete / tasks you were not able to complete.

The deadline for submission will be **11.59 PM (IST).** Late submissions will not be accepted or evaluated under any circumstances.

Best of luck!