GoldPulse – My Zense Recruitment Project

Motivation behind the project

The first and foremost motivation behind this project came from me looking for a personal finance management app on play store. I found some apps like "Splitwise, Write more names", but none of these apps gave me proper money management. Some of them may show monthly budget, but didn't gave any idea about how to manage my money if I wanted to go out with friends to Goa after my endsem. At the same time, I also wanted to know how much can I afford to spend in Goa, if I wanted to buy a phone by the end of next Sem as buying phone has a higher priority for me . Also, I saw this is problem faced by many of my friends too. Also, my father was looking for a place where he can manage his income from various sources allowing him to know where his earning is growing by a rate which is more than inflation. He also, wanted to get reminded about payments and if possible, to pay it automatically. So, I decided to make a website which incorporates above features. I thought why not to make this as full – fledged personal finance management website which not only incorporates the above features, but also gave timely investment tips, keep a track of my investments and manage earning from them.

Models:

1. Accounts – This app is used to manage user authentication. The model Account stores user details.

   Features:

   - Login, Logout, And Signup
   - Login using either email or username
   - Reset password using email link verification
   - New user account verification using email link
   - Link validity upto 4 hours

2. Balance – This model is used to keep track of user balance, how should he divide his earning between investing, and expenses.

Features:

- I have implemented cron job kind of thing using Scheduler which is used to keep track of user expense in expected coming month and investments in next year. The function runs twice a day but its values are only updated at start of month. This is done to give time to income to relax over time and not show rapid increase in allowed expenses thus managing user expense properly.
- I have allocated (70% of Last Month Income) in Coming month expenses (Needs and Wants) and rest 30% is allocated in Investments.

3. Income Model

Features:

- Add income source to balance. I have given choice to the user to either automatically sync his income or he can add manually after every chosen frequency time. If user chooses to automatically sync his income, then at every time income will be added and user will be notified by an email.
- Income summary and View - Analysis of income over years, its growth vs inflation. Charts given for better understanding. (Used JsonResponse for sending data to charts)
- Delete and edit income
- Analysis of each source over time, growth rate, inflation, contribution to total income. Charts given.
- Inflation data scraped from https://www.macrotrends.net/countries/IND/india/inflation-rate-cpi

4. Expense

Features:

- This app is sub-divided into three models
  - Expense – This model is used to keep track of user's allowed expenses, expenses in wants, needs, buffers stored from previous months. It's instance is created every month using cron work.

    - Wants – This model is used to keep track of wants of the user. It's instance is created for every wants.
    - Needs – This model is used to keep track of needs of the user. It's instance is created for every wants.
- Wants:
    - If user has a want to buy an item in less than 30 days and he doesn't have enough money in this month's wants section, then he will be redirected to the same page with an additional option enabled saying that if he wants to use his buffer amount.
    - If user has a want to buy an item which is more than 30 days, then user pays min(wants left per month according to its priority, per month as buy date). If at any time, his account balance or even his allowed amount in wants in exceeded then he will be notified by email. At buy date, user will be notified if his saved amount is enough or not.
    - User at buy date can deny buying this item and can either add money to fund other wants, withdraw to increase fund of expense or investment. Only after user clicks buy, he withdraws this saving completely.
    - Wants per month funds are added from allowed wants in order of their priority which can be changed by the user.
- Needs:
    - Besides basic CRUD and cron operations, if user's currently allowed needs money is low than amount then :
        - If the item's buy frequency is more than 91 days or it is a one-time purchase (I have done this because items bought quarterly, biannually or annually have generally higher prices) then user has a right to use his money saved in buffer to pay for this item.
        - If the need is an emergency, then without checking any budget conditions, item is bought.
        - Same is considered when user chooses to automate his bills using cron.
- Analysis provided with charts, wherever needed.

5. Investments – This app is sub-divided into the following models:
    - Investments- This model in used to keep track of user's investments in various asset classes which is updated based on user's age, income and previous choices chosen while selling items or withdrawing money from wants_items. It tells user how much to invest in each asset class in a year. (I have used x:100-x approach for safe:risky assets)

    - StockData- I have made this model to reduce the loading time. Basically, I am giving users stock recommendation and analysis using data scraped from "Moneyworks4me" website, yfinance, dhan, finance.yahoo, for various indicators and parameters like P/E , P/B, DOE, EV/EBDITA, etc. which I combined and gave user a final rating. I have calculated RSI and combined it with 52 weeks high and low prices and gave buy signals are given based on these values. I have also considered the if a stock is in any indices like Nifty50 ,Nifty Pharma, etc. or not, and included that factor in ratings. To give short term buy signal, I have used a module named "tradingview_ta". I have given user stock predictor which tells how good is a stock based on the above parameters. To get lists of all traded stocks and their symbols, I have scarped data from NSE (National Stock Exchange Website) and stored it in a csv for making it fast.

    - Stock – It is used to keep track of position, previous holding, P/L, etc. of a stock bought by the user. While selling the stock, I have kept track of FIFO model which is used in stock exchanges. Upon selling the stock, I have given user the option of either including the amount in Expense or in the Investments. I have kept track of stock recommendation based on 40:15:10 in Large, Mid and Small cap. While, buying or selling I have kept track of current market prices from yfinance module.

    - MFData – This model works in a similar way as StockData but for Mutual Funds. I have scraped data from websites like

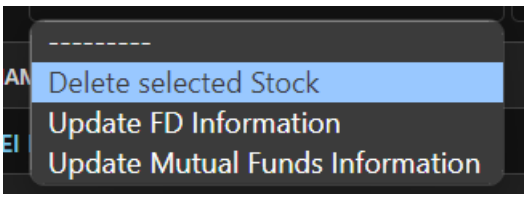
"etmoney" and gave an estimated rank. I am giving recommendations to users based on this data. I have then used ARIMA (Autoregressive integrated moving average) model to give user price recommendation for next 5 days, min and max within a month. To get symbols against names of Mutual funds, codes.json is maintained.

- MF – This model is used to keep track of Mutual Funds bought by the user. If a user buys a SIP, then each month using cron, his payment is made. When a user sells a SIP, he is given choice to either add this money to expense or to increase his investment.

- Other – This model is used to keep track of FD. User can invest money, break it before maturation, or it is automatically sold at maturation.

- SGB – It is used to keep track of Sovereign Gold Bonds, held by the user. The gold bonds can be bought for a period of 5 , 8 or 11 years. During this period, 2.5% simple interest per year using cron.

I have given permission to the admin to update MFData and sync FD maturation details . These are also auto updated using cron job.



TEMPLATES – This folder keeps track of all html templates.

STATIC – This folder keeps track of static resources like css, images, js used.

Logging is done to allow admin to know if any error has occurred. Log for cron work is specially maintained in cron_log.log file.

Problems Faced:

- Figuring out how to do the cron job part was toughest for me. I tried about 10-12 answers from stackoverflow, github , medium, but none worked for me. Earlier I was doing time.sleep() but it suspended the main page. With schedular, even if cron faces an error, the main page doesn't crash.
- Another problem faced was figuring out how to do chart.js part without REST.
- I faced many small glitches in bs4 or Django or ARIMA, but that I easily resolved using the answers on stackoverflow.

Future Ideas:

1. Easy Difficulty Ideas –
    a. Django messages sometimes not work as expected
    b. Notification model which shows notification and updates (even sent on email) whenever user logs in.
2. Medium Difficulty Ideas –
    a. Building good frontend using React
    b. Adding News API for giving recommended news based on his investments and expenses
    c. Adding features of Insurance, Emergency Funds to the current project.
3. Hard Difficulty Ideas –
    a. Asking for permission from banks, UPI, demat accounts, etc. to make this project work with real money.
    b. Giving recommended buying sources or sites with cheaper prices when user is buying an item.
    c. Creating API Endpoints so that this project can be used as a backend for a developer building mobile apps.
    d. Using celery beat and redis server instead of Scheduler for performing automated tasks.