naugets

FULL STACK ASSESSMENT

The task is to develop a feature for a Gold Fintech app to calculate a user's portfolio (Profit and Loss) using an API. The feature should be intuitive, user-friendly, and secure, allowing only authorized users to access their portfolio details, including gold weight, current value, and related information.

Gold Fintech App is a digital platform for buying and selling gold through a mobile app, making it convenient and accessible to a wider range of investors. Users can purchase gold in small denominations, track their investment performance in real time, and make trades directly within the app. The app allows users to buy and sell gold, and their portfolio's value changes depending on the current gold price. If the gold price increases the portfolio will be in profit otherwise in a loss.

The app's code base includes creating a wallet transaction record with a "CREDIT" type when a user adds funds and a "DEBIT" type when they withdraw funds. Here is the wallet transaction schema thus formed:

```
{
        userId: { type: ObjectId, required: true, ref: 'User' }, // To store the user id
        amount: { type: Number, required: true }, // Amount of the transaction done.
        type: { type: String, required: true, enums: ['CREDIT', 'DEBIT'] }, // Type - debit or credit.
        status: {
          type: String,
          required: true,
          enums: ['FAILED', 'SUCCESS', 'PROCESSING'],
        }, // Status of the transaction being done.
        runningBalance: { type: Number, required: true }, // Running Balance of the user after each
                transaction.
        transaction: { type: ObjectId, ref: 'Transaction' }, // Gold transactions reference.
        createdAt: { type: Date, required: true }, // Created At date
        updatedAt: { type: Date, required: true }, // Updated At date
}
```

The credited / Debited amount is also reflected in the user's running balance. The running balance is stored in the user's collection. You can assume the user's schema as :

```
{
firstName: { type: String, required: false },
lastName: { type: String, required: false },
password: { type: String, required: false },
mobileNumber: { type: String, required: true },
country: { type: String, required: true },
email: { type: String, required: true },
runningBalance: {
  wallet: { type: Number, required: true }, // CURRENT FUNDS STORED
  gold: { type: Number, required: true }, // CURRENT GOLD QTY IN GMS
  goldPrice:  { type: Number, required: true }, // CURRENT GOLD PRICE
  }
}
```

Based on the funds available in a user's account, they can buy or sell gold through the app. When a user buys gold, a transaction record is created in the gold transaction schema with the type "CREDIT". At the same time, a record of type "DEBIT" is created in the wallet transaction collection. Similarly, when a user sells gold, a transaction record is created in the gold transaction schema with the type "DEBIT", and a record of type "CREDIT" is created in the wallet transaction collection for the specific amount of gold being purchased or sold.

The gold transaction schema includes the following fields:

- userId: A required object ID reference to the user.
- entityUser: An object ID reference to the user entity.
- quantity: A required number indicating the quantity of gold in gms.
- amount: A required number indicating the amount spent or earned.
- type: A required string with possible values of "CREDIT" or "DEBIT".
- status: A required string indicating the transaction status, with possible values of "FAILED", "SUCCESS", "WAITING", "CANCELED", or "PENDING".
- runningBalance: An object with required number fields indicating the wallet, loyalty points, and gold balances.
- createdAt: A required date indicating when the transaction was created.
- updatedAt: A required date indicating when the transaction was last updated.

Given that Current Gold Quantity is in grams and the current amount balance is also stored in the running balance of the user schema provided.

Based on all three schemas provided for wallet transactions, gold transactions, and user collection. You have to write an API that will use data needed from all these collections and should be able to provide the user's overall growth or loss in response.

Response should be JSON:

```
{
 “netFundAdded”:‘any amount’,
 “currentFund”:‘any amount’,
 “netGrowthOrLoss”:‘any amount’, //If growth it should be (+)ve otherwise (-)ve.
 “gainOrLossPercentage”: ‘any percentage value’
 }
```