MateMachine™

Hiring Task

1. The Currency Converter

Write a configurable module that given a list of currency exchange rates, converts currencies to one another. Let's carry on with an example. Assume we have the following conversion rates:

- (USD => CAD) 1.34
- (CAD => GBP) 0.58
- (USD => EUR) 0.86

If we want to convert USD to EUR it's pretty straight forward, we just multiply the amount by the conversion rate which is 0.86. Also, to convert CAD to USD we may divide the amount by the conversion rate which is 1.34. On the other hand, notice that if we want to convert CAD to EUR there's no direct way to do so. Our currency converter should be able to do this by finding a "conversion path", if you will. The conversion path requires our converter in order to convert CAD to EUR to first convert the amount to USD, and then to EUR. Implement *ICurrencyConverter* with the following requirements in mind:

- The converter should find the shortest conversion path (if any).
- The converter is a singleton with potentially multiple threads invoking its *Convert* method.
- Because the *Convert* method is frequently invoked, optimization is top priority in terms of minimum locking, less processing cost etc.

Solve this problem considering a practical production environment.

```
interface ICurrencyConverter
{
    /// <summary>
    /// Clears any prior configuration.
    /// </summary>
    void ClearConfiguration();

    /// <summary>
    /// Updates the configuration. Rates are inserted or replaced internally.
    /// </summary>
    void UpdateConfiguration(IEnumerable<Tuple<string, string, double>> conversionRates);

    /// <summary>
    /// Converts the specified amount to the desired currency.
    /// </summary>
    double Convert(string fromCurrency, string toCurrency, double amount);
}
```

2. The Social Network

What we need you to do for this task is to make a small .NET Core MVC web application. Only the backend matters; the frontend can be simple text and it doesn't matter at all. The requirements are as follows:

- Users must authenticate (register + login). It doesn't matter how you do this, it's just a natural requirement.
- Users may view a list of all users, searchable by name.
- A user may request to become friends with another user, or unfriend. Like Facebook, the friendship is mutual and is requested first.

The following are what matters for this task:

- Design of the database schema, avoiding data redundancy
- Optimization of SQL queries and operations, achieving better performance
- Details of this task are up to you, for example you can match user names literally, or consider typos.