1. Problem - revenue recognition problem

Our application will revolve around a problem related to finances domain usually referred to as **"revenue recognition problem"**.

Revenue recognition is a common problem in business systems. It’s all about when you can actually count the money you receive on your books. If I sell you a cup of coffee, it’s a simple matter: I give you the coffee, I take your money, and I count the money to the books that nanosecond.

For many things it gets complicated, however. Say you pay me a retainer to be available that year. Even if you pay me some ridiculous fee today, I may not be able to put it on my books right away because the service is to be performed over the course of a year. One approach might be to count only one-twelfth of that fee for each month in the year, since you might pull out of the contract after a month when you realize that writing has atrophied my programming skills.

The rules for revenue recognition are many, various, and volatile. Some are set by regulation, some by professional standards, and some by company policy. Revenue tracking ends up being quite a complex problem.

In real life, improper revenue recognition has been at the heart of several major corporate scandals, such as Enron and WorldCom. These companies used various tactics to misrepresent their financial health, leading to significant legal consequences and financial losses for investors. Thus, accurate and compliant revenue recognition is essential for maintaining transparency and trust in financial markets.

Additional fact: Enron was the company Evil Corp from Mr. Robot was inspired by

2. Context

We are creating a REST API application - Revenue Recognition System for a large corporation ABC.

3. Functional requirements

To develop a robust Revenue Recognition System, we need to ensure that it accurately recognises revenue for various types of products. We should also allow the user

3.1. Managing clients

Use cases:

1. add client,
2. remove client,
3. update data about client

We would like to store information about clients, who may be either individuals or companies. For individuals, we need to store the first name, last name, address, email, phone number, and PESEL (Polish National Identification Number). For companies, we need to store the company name, address, email, phone number, and KRS number (National Court Register number). All the mentioned data are required.

We need the ability to add, update, and remove clients. For individual clients, the PESEL number cannot be changed once it is entered. Similarly, for companies, the KRS number cannot be changed once it is entered.

When removing data about an individual client, we perform a soft delete. This means we overwrite the data in the database, but keep the record itself in the database. Data about companies cannot be removed.

3.2. Software license

Our client is selling different kind of good and services. One of the things sold are software systems. Each software system has a name, description, information about current version and category (finances, education etc.).
Each software may be sold either on subscription basis and/or an upfront cost.

3.2.1. Discounts

There are discounts associated with products. Discounts may be applied to the upfront cost or a subscription. Discounts are expressed as a percentage and have a specified time range during which they are active.

For ex.
Name Offer Value. Time range
Black Friday Discount. Discount for subscriptions 10%. 01-01 to 03-03

The discount is only applied to contracts created or subscriptions created during the discount's time range.

3.2.1. Upfront cost

Use cases:**
4. create contract
5. issue payment for the contract

For a specific software, we should know exactly what the cost of buying a license for this software is for one year.

If we have a client who wants to buy our software using the upfront approach:

Creating the contract:

- We first create a contract for the client. The contract has a **start date and an end date**. The time range should be **at least 3 days and at most 30 days**. The contract **has to be paid for by the client** within this time range. Otherwise contract is cancelled.

- The contract also has a **price associated with it.** The price includes **all possible discounts.**
- The contract contains information about **updates** that the client can receive for the bought software.
- Each contract provides **at least 1 year of updates** to whoever bought the product. We can offer **additional support** within the contract—**each additional year adds an extra 1000 PLN to the contract cost.** We can only extend the support by 1, 2 or 3 years,
- When **multiple discounts** are available, we choose the **highest one**.
- All **previous clients of our company receive a 5% discount** for returning clients. A previous client means that the client has bought at least a single subscription or signed a single contract. This discount may be added to the other ones.
- The contract is associated with the chosen software and a specific version of that software.
- Each contract **specifies the version of the software** associated with it.
- The contract **cannot be changed once created**; it can only be removed.
- Creating the contract **does not mean that the client has signed it.**
- The **price on the contract cannot be treated as revenue.** Only after issuing a full payment we can treat the value on the contract as revenue.
- When creating the contract, ensure that the client **does not already have an active subscription or an active contract for this product.**

After preparing a contract for the client we are waiting for the client to pay for the contract. Client has to issue a full payment within the time range specified in the contract.

Paying for the contract:**

- The client can pay for the contract. Payments can be **made either as a single payment or in multiple installments.**
- The **total value of all payments must equal the amount listed in the contract.**
- Once the client has paid the full amount for the contract, **we assume that the contract is signed.**
- After the contract is signed, **we can treat the contract value as revenue.**
- We cannot accept the payment for a contract after the date specified within the contract. If the client was late - we will have to create another offer for him. The previous payments will be returned to the client and cannot be treated as revenue.

3.2.2. Subscriptions - optional

The subscriptions part is optional. You can implement it if you have enough time.

Use cases:
6. buying a subscription by the client
7. paying for the subscription

Our company also offers subscriptions for using specific software services.

- Each subscription is associated with a **single client** (individual or company).
- Each subscription is associated with a **single software service**.
- Each subscription offer for a software has:
  - **Name**
  - **Renewal period (**monthly, yearly, etc.)
  - **Price** that has to be paid at the beginning of each renewal period
  - The renewal period should be **at least 1 month and up to 2 years.**

Buying a new subscription:**

- We register a new subscription sale for a specific client and specific software.
- We must include the necessary information about the subscriptions.
- We assume that registering the subscription means the client **has already paid for the first renewal period. We can treat this as our revenue.**
- When calculating a payment, **remember to include a 5% discount for our loyal clients.** This discount is applied to all the renewal payments.
- Other discounts are applied only **if they are active during the registration of new subscriptions.** We apply the highest discount to the price for the first renewal period. **We apply the discount only to the first payment.**
- We apply additional 5% discount to our loyal clients.

Paying for the subscriptions:

- The client should pay for the subscription at the beginning of the new renewal period. If the user hasn't already paid for other renewal periods - we cancel the subscription.
- The system **should not accept the payment** if the client has already paid for current renewal period. We **only accept the payment for the current renewal period within the time range of this period.**
- If the amount paid by the customer is not equal to the renewal period amount, we return an exception.
- We apply the loyal customer discount for the renewal payment.

3.3. Revenue calculation

Use case:
7. Calculate revenue
8. Calculate predicted revenue

We would like to be able to calculate the current revenue for the company. Specifically, we need the ability to calculate revenue (those may be additional parameters when sending the request about revenue calculation):

- For the entire company.
- For a specific product.
- For a specific currency. We can assume that all the currency data in the database is stored in PLN. Find a public service that returns the exchange rate for a specific currency. Use this rate to recalculate the revenue into the required currency.

We would like to be able to calculate both our current revenue and predicted revenue. The difference is that current revenue takes into account only the money received from payments and contracts.

For predicted revenue, we can calculate the value based on the following assumptions:

- Users will continue paying for their subscriptions.
- All contracts that are not yet signed will eventually be signed.

Technical requirements

- Remember to design the architecture properly.
- Try to create a domain model, you can reuse your domain model and use EF at the same time.
- Prepare set of unit tests used for testing the business logic.
- Use the Entity Framework.
- Design the proper REST API endpoints.
- Make sure that we can test the API using Swagger.
- Use all the good practices and design patterns you think are useful.
- You can either separate your application using directories (namespaces) or separate projects.
- You can use either the "vertical" or "horizontal" approach.
- **All the endpoints should only be accessible to the logged in Employee. We have to know only the login, password and the role of the employee. We assume that only editing and removing clients is accessible to users with role admin. The rest of the use cases is accessible to standard user.**
- **Even if you won't manage to implement all the functionalities - upload the project.**